Wil van der Aalst
Kees van Hee

Tradução: Jorge Cardoso

Gestão de Workflows

Modelos, métodos e sistemas

IMPRENSA DA UNIVERSIDADE DE COIMBRA
COIMBRA UNIVERSITY PRESS

• COIMBRA 2009

(Página deixada propositadamente em branco)

E N S I N O

IMPRENSA DA UNIVERSIDADE DE COIMBRA
COIMBRA UNIVERSITY PRESS

EDIÇÃO
Imprensa da Universidade de Coimbra
Email: imprensauc@ci.uc.pt
URL: http://www.uc.pt/imprensa_uc
Vendas online: http://siglv.uc.pt/imprensa

CONCEPÇÃO GRÁFICA
António Barros

EXECUÇÃO GRÁFICA
Tipografia Lousanense, Lda.

ISBN
978-989-26-000-0

DEPÓSITO LEGAL
.....

Wil van der Aalst
Kees van Hee

Tradução: Jorge Cardoso

Gestão de Workflows

Modelos, métodos e sistemas

IMPRENSA DA UNIVERSIDADE DE COIMBRA
COIMBRA UNIVERSITY PRESS

• COIMBRA 2009

Prof.dr.ir. W.M.P. van der Aalst
Eindhoven University of Technology
Faculty of Technology and Management
Department of Information and Technology
PO Box 513
NL-5600 MB Eindhoven
The Netherlands
w.m.p.v.d.aalst@tm.tue.nl

Prof.dr. Kees Max van Hee
Eindhoven University of Technology
Faculty of Mathematics and Computing Science
PO Box 513
NL-5600 MB Eindhoven
The Netherlands
kvanhee@deloitte.nl

Sumário

(Página deixada propositadamente em branco)

PREFÁCIO

A perspectiva comum sobre sistemas de informação, que é constituída por aplicações de base de dados feitas sob medida e com um custo elevado, está a mudar rapidamente. A mudança é incentivada em parte pelo amadurecimento da indústria de *software* que tem feito um melhor uso de componentes genéricos e soluções *standard* de *software* e pelo investimento na revolução da informação. Por outro lado, esta mudança tem resultado num novo conjunto de pedidos de serviços de informação que são homogéneos nos seus padrões de apresentação e interacção, abertos na sua arquitectura de *software* e globais na sua abrangência. Os pedidos chegaram maioritariamente de domínios de aplicação tais como o comércio electrónico e operações bancárias, indústria (incluindo a própria indústria de *software*), formação, educação e gestão do ambiente, entre outros.

Os futuros sistemas de informação terão de suportar uma boa interacção com uma grande variedade de recursos de dados de vários vendedores e independentes, e aplicações herdadas de gerações de sistemas de informação anteriores que se executam em plataformas heterogéneas e em redes de informação distribuídas. Os metadados irão desempenhar um papel crucial na descrição do conteúdo dessas fontes de dados e na facilidade da sua integração.

Uma grande variedade de padrões de interacção orientados para a comunidade terá de ser suportada pela próxima geração de sistemas de informação. Tais interacções poderão envolver a navegação, resposta a pergunta e obtenção de dados e terão de ser combinadas com notificações personalizadas, anotações e mecanismos de perfil. Essas interacções também deverão ter interfaces construídas de forma inteligente com aplicações de *software* e necessitarão de ser dinamicamente integradas em ambientes personalizados e cooperativos com um elevado grau de integração. Além disso, os fortes investimentos em recursos de informação pelos governos e empresas requerem medidas específicas que garantam a segurança, a privacidade e a exactidão dos seus conteúdos.

Todos estes são desafios para a próxima geração dos sistemas de informação. Chamamos a estes sistemas *Sistemas de Informação Cooperativos*, e são o centro de interesse desta série de publicações.

Em termos leigos, os sistemas de informação cooperativos satisfazem um conjunto misto de requisitos caracterizados por *conteúdo – comunidade – comércio*. Estes requisitos originam a necessidade de encontrar soluções de *software* personalizadas, tais como os sistemas de planeamento de recursos empresariais (ERP – *Enterprise Resource Planning*) e os sistemas de comércio electrónico.

Um dos maiores desafios na construção de sistemas de informação cooperativos é o desenvolvimento de tecnologias que permitam um aumento e evolução constante dos fortes investimentos actuais nos recursos e sistemas de informação. Tais tecnologias deverão oferecer uma infra-estrutura apropriada que suporte não só o desenvolvimento, mas também a evolução do *software*.

Os resultados recentes de pesquisas em sistemas de informação cooperativos estão a tornar-se a tecnologia base para a criação de portais de informação orientados à comunidade. Um portal de informação fornece um lugar de compra único para uma grande quantidade de recursos e serviços de informação, criando assim uma comunidade de utilizadores fiel.

Os avanços da investigação que dão origem a sistemas de informação cooperativos não advêm de uma só área no campo das tecnologias de informação. Bases de dados e sistemas de conhecimento, sistemas distribuídos, *groupware* e interfaces gráficas com o utilizador, todas estas tecnologias também amadureceram. Enquanto melhorias adicionais para tecnologias individuais são desejáveis, a maior vantagem para o avanço tecnológico é esperado vir da sua evolução numa tecnologia adequada para a construção e gestão de sistemas de informação cooperativos.

A série de publicações intitulada *MIT Press Cooperative Information Systems* cobre esta área através de livros didácticos e edições de pesquisa para investigadores e profissionais que desejem continuar actualizados sobre os novos desenvolvimentos e as tendências futuras.

Esta série apresenta três tipos de livros:

- Livros didácticos ou livros de referência para alunos do 2º ciclo universitário ou para disciplinas de pós-graduação;
- Monografias de investigação que recolhem e resumem resultados de pesquisas e experiências de desenvolvimento ao longo de vários anos; e
- Volumes editados, incluindo colecções de artigos sobre um tópico específico.

Os autores são convidados a apresentar propostas para novos livros que incluam uma tabela de conteúdos e uma amostra de capítulos. Todas as submissões serão formalmente examinadas e os autores receberão uma resposta à sua proposta.

John Mylopoulos
University of Toronto

Michael Papazoglou
Tilburg University

Joachim W. Schmidt
Techniche Universität TUHH

AGRADECIMENTOS

Este livro foi preparado em cooperação com os grupos de *workflow* da Deloitte & Touche Bakkenist, a Faculdade de Matemática e Ciências da Computação e a Faculdade de Gestão de Tecnologia da Universidade de Tecnologia de Eindhoven. Os autores gostariam de agradecer a todos os membros (actuais e passados) e estudantes destes grupos, em particular, Twan Basten, Silvia de Gast, Ernst Kleiberg, Selma Liman, Michael van Osch, Jaap Rigter, Eric Verbeek, Mark Voorhoeve, Laurens Vrijnsen, Gerd Wagner e Jaap van der Woude. Queríamos também agradecer ao Michiel Bos e ao Niels van Riel por nos ajudarem a preparar a versão em inglês do nosso livro e à Monique Jansen por fazer a revisão final do livro.

Um agradecimento especial para os nossos co-autores, André Blommers e Peter van der Toorn, que contribuíram, cada um deles, com um capítulo. Por último, queríamos agradecer à Dutch Tax Authority pela permissão para usar o projecto Sagitta 2000 como um caso de estudo apresentado neste livro.

Dezembro 2000
Wil van der Aalst
Kees van Hee

(Página deixada propositadamente em branco)

INTRODUÇÃO

Este livro trata da gestão de processos de negócio. Certamente não se trata de um novo tópico. Desde o início da Revolução Industrial, se tem escrito sobre essa temática sob vários pontos de vista – económico, social, psicológico, contabilístico, engenharia mecânica e administração de negócios. Neste livro, examinamos a gestão dos processos de negócio na perspectiva da computação, ou – de uma forma mais ampla – da tecnologia de informação. A razão para tal é que a tecnologia de informação tem dado passos gigantescos nos últimos anos, resultando na criação de formas totalmente novas de organizar os processos de negócio. O desenvolvimento de pacotes de *software* genérico para a gestão de processos de negócio – chamados de sistemas de gestão de *workflow* (*Workflow Management Systems* – WFMS) – é particularmente importante neste aspecto.

Até bem recentemente, a regra de ouro era: "Primeiro organizar, depois computorizar." Isto implicou que os processos fossem desenvolvidos com a suposição implícita de que o processo de negócio fosse maioritariamente gerido por pessoas. Então, uma estrutura organizacional seria desenvolvida representando os grupos de pessoas, ou departamentos, que eram associados a tarefas específicas. Apenas depois, as pessoas começaram a considerar que os computadores – ou melhor dizendo, os sistemas de informação – podiam parcialmente suportar, ou mesmo gerir o trabalho. Esta aproximação não examina suficientemente as oportunidades proporcionadas pelos sistemas de informação. Neste momento atingimos um ponto de transição: primeiro desenhamos os processos de negócio de uma forma mais abstracta, sem considerar a implementação, e depois desenhamos os sistemas de informação e a sua organização passo a passo. De facto, decidimos que cada tarefa de um processo deverá ser realizada por um sistema de informação ou por uma pessoa.

No entanto ainda existem alguns problemas com esta descrição. Primeiro, a noção de que podemos organizar processos de negócio de forma diferente usando sistemas de informação, não é um novo conceito. Há muito tempo que as pessoas têm vindo a fazer isso com os processos de negócio cujo principal objectivo é o processamento de informação. Durante os anos 70 foram realizados esforços para computorizar a gestão dos processos de negócios usando sistemas de informação. Isto provou ser impossível com a tecnologia disponível até então. Mesmo hoje em dia, e num futuro mais próximo, existem e continuarão a existir muitas tarefas de processos de negócio que podem apenas ser realizadas por pessoas. Em reacção às tentativas descuidadas dos anos 70, o papel desempenhado pela tecnologia de informação tem sido de certa forma restrito.

Os sistemas de informação são usados para reduzir a quantidade de trabalho por pessoa, particularmente nos escritórios. Analisando minuciosamente o que

as pessoas fazem nos seus gabinetes – perguntando porque o fazem – foram identificadas as seguintes funções de processamento de informação: escrita de texto, desenho, cálculo, arquivo e comunicação de informação. A análise destas actividades levou ao desenvolvimento dos seguintes produtos: processadores de texto, sistemas de desenho, sistemas de folhas de cálculo, sistemas de base de dados e sistemas de correio electrónico. Todos estes sistemas são de natureza genérica: não estão limitados a uma área específica do negócio – como são, por exemplo, os sistemas de contabilidade – e por consequência são vastamente usados. Graças à distribuição em massa, estes sistemas de *software* são de alta qualidade e relativamente baratos. (De facto, os sistemas de contabilidade são bastante utilizados, mas não tão extensivamente como processadores de texto).

Em parte devido a este desenvolvimento, o impacto das tecnologias de informação cresceu rapidamente, e por sua vez levou várias pessoas a estudarem as novas possibilidades apresentadas. Este estudo resultou no "BPR Wave". O BPR (*Business Process Re-engineering*) significa a reengenharia de processos de negócio e é um método para melhorar a eficácia e a eficiência dos processos de negócio. O BPR é baseado na noção de que, se é feito um uso extensivo de tecnologias de informação, então os processos de negócio poderiam ser inteiramente diferentes daqueles que encontramos hoje em dia. É por isso acertado que se reestruture completamente os processos actuais, do modo descrito anteriormente. A organização dos processos de negócio deixa de ser assim privilégio do perito organizacional ou de negócio: o tecnologista de informação tem agora um papel importante a desempenhar. Este facto é positivo porque o tecnologista é uma pessoa encarregue de desenvolver processos com um certo grau de excelência. Afinal de contas, cada algoritmo define um processo. Contudo, até recentemente, o papel do tecnologista de informação era limitado ao processamento da informação nos sistemas de computadores – mas na realidade a tarefa principal de muitos outros processos de negócio é o processamento de informação.

No passado, era a estrutura funcional de uma organização que desempenhava o papel mais importante na forma como estava organizada. Hoje em dia, os processos de negócio são cruciais. É necessário escolher uma boa ferramenta de referência para que os processos possam ser definidos e analisados com clareza. A definição é importante quando preparamos um (re)estrutura e antes de decidirmos implementar um novo processo é importante primeiro estabelecer se esse processo irá funcionar correctamente. Para tal, é necessário ser capaz de analisar o processo definido. Isto pode ser feito de várias formas. Por exemplo, métodos formais podem ser usados para identificar as propriedades dos processos, ou a falta delas. Outro método de análise utiliza técnicas de simulação, muitas vezes suportadas por animações gráficas. Para isso, são fundamentais ferramentas de *software* de suporte a essas actividades.

Este livro apresenta uma ferramenta de referência para definir processos e discute métodos analíticos. Desta forma, é feito um uso extensivo de redes de Petri, um conceito formal que tem sido desenvolvido desde a década de 60 e que sofreu avanços significativos na década de 80. As redes de Petri são bastante adequadas para definir e analisar processos complexos. Outra propriedade importante é o facto de tornar a definição dos processos fácil de entender às pessoas. Isto facilita a comunicação entre *designers* e utilizadores. Existem também ferramentas de *software* que suportam a definição e a análise de processos.

A partir do momento em que novos processos de negócio são desenvolvidos, estes têm de ser implementados. A gestão e, em parte, a execução dos processos são controladas por pessoas com a ajuda de sistemas de informação. Tal como já foi referido, durante os últimos anos, uma nova categoria de *software* genérico foi evoluindo: os sistemas de gestão de *workflow*. Este *software* suporta os processos de negócio tendo em conta a sua logística de informação. Por outras palavras, os sistemas de gestão de *workflow* garantem que é dada a informação correcta à pessoa certa, na altura adequada, ou é submetida para a devida aplicação informática, no momento certo. Um sistema de gestão de *workflow* de facto, não realiza qualquer das tarefas de um processo. A força de um sistema de *workflow* reside no facto de ser um *software* genérico que pode ser usado em várias situações e cenários. A sua fraqueza deriva de geralmente ser necessário recorrer a aplicações de *software* para executar os processos de negócio.

O termo "workflow" é utilizado aqui como um sinónimo de "processo de negócio". Iremos utilizar, sempre que possível, a terminologia desenvolvida pelo *WorkFlow Management Coalition* (WFMC). Esta é uma organização dedicada ao desenvolvimento de terminologias *standard* e interfaces padrão para os componentes dos sistemas de gestão de *workflow*.

Este livro começa por descrever a organização dos *workflows*. Isto é importante para perceber o papel que os sistemas de gestão de *workflow* podem desempenhar e como deverão ser utilizados. Os termos que são requeridos para lidar com estes processos são introduzidos de um modo informal, fornecendo assim as bases para o resto do livro. Depois segue-se um capítulo sobre a modelação de *workflows* que inclui uma introdução simples à teoria das redes de Petri. O capítulo seguinte aborda a gestão dos recursos que contribuem para os processos de negócio. Esses recursos podem ser pessoas, mas também podem ser máquinas ou computadores. Serão consideradas também as técnicas para a análise de processos. Seguidamente são introduzidos os sistemas de gestão de *workflow*, com uma abordagem às suas funções e à sua arquitectura. Depois é apresentada uma metodologia para o desenvolvimento de *workflows*. O último capítulo é dedicado a um caso de estudo de uma aplicação real.

No apêndice, incluímos um glossário organizado por ordem alfabética contendo todos os termos relevantes utilizados, juntamente com seus sinónimos e uma breve definição. A primeira vez que um termo importante for utilizado será escrito em itálico.

Este livro é destinado a estudantes de tecnologias de informação, engenharia industrial e para aqueles que se encontram profissionalmente envolvidos na implementação de BPR usando os WFMS.

(Página deixada propositadamente em branco)

Capítulo 1.

ORGANIZAR WORKFLOWS

1.1 Ontologia da gestão de workflows

O objectivo deste capítulo é o desenvolvimento de uma ferramenta de referência. Esta ferramenta tem três funções neste livro. Primeiro, é usada para definir o contexto da gestão de negócio, sob o qual operam os sistemas de gestão de *workflow*. Segundo, é usada para modelar e analisar processos. Por fim, é utilizada para descrever a funcionalidade e a arquitectura dos sistemas de gestão de *workflow*. Uma ferramenta de referência é um sistema de expressões bem definidas que descrevem uma área particular de conhecimento. É também designada e conhecida por ontologia.

A ontologia em que estamos interessados é a dos processos. Os termos utilizados são de natureza genérica e podem ser aplicados virtualmente em todas as situações de trabalho. No entanto, na prática, existem vários sinónimos que são amplamente utilizados; por questões de clareza, tentaremos usar um único "termo preferido" sempre que for possível. Os termos adoptados estão de acordo com a terminologia utilizada pelo Workflow Management Coalition. Neste capítulo, discutiremos primeiro o papel do trabalho na sociedade, depois examinaremos os processos, seguidos da distribuição de trabalho. O relacionamento entre o supervisor e o fornecedor desempenha aqui um papel importante. Estes relacionamentos são extremamente importantes especialmente nas interacções de negócio electrónicas. Seguidamente, estudaremos as estruturas organizacionais e a gestão de processos. Finalmente, analisaremos o papel dos sistemas de informação (computorizados) no estabelecimento e na gestão dos processos de negócio.

1.2 Trabalho

As pessoas trabalham para poder viver – mesmo parecendo que alguns envolvem-se demasiado com o trabalho dando a impressão que vivem para o trabalho. De facto, trabalhamos porque precisamos de produtos para nos mantermos vivos (por exemplo: comida, roupa, uma casa, um meio de transporte, para não falar do entretenimento). Não produzimos, por nós próprios, todas as coisas que precisamos pois isso seria ineficiente. Na realidade, numa sociedade moderna seria impossível produzir todos os produtos que usamos na nossa vida, por nós próprios. Teríamos de aprender muitas técnicas

diferentes e complexas e para tal demoraríamos a vida inteira. Precisaríamos de muitas vidas só para fazer as ferramentas necessárias para produzir as necessidades da vida. É por isso que em vez disso estamos organizados em "processos de negócio" especializados, nos quais as pessoas produzem uma gama limitada de produtos de forma altamente eficiente, com a ajuda de máquinas. Estes produtos são fornecidos a outras pessoas através de um mecanismo de mercado e uma estrutura de distribuição em troca de dinheiro, o que proporciona aos fabricantes a compra de produtos, não fabricados por eles. Com a produção distribuída desta forma, é criado trabalho que não existiria se todos nós fôssemos auto-suficientes na criação dos produtos que precisássemos. Por exemplo, a gestão de dinheiro – o que os bancos fazem – e preparar material de publicidade não seriam necessários.

Desta forma têm sido desenvolvidos vários tipos de serviços e produtos que não têm uma contribuição directa na nossa sobrevivência, mas que são necessários para manter a organização operacional. Apesar deste "fardo", somos capazes de produzir de uma forma tão eficiente que temos uma grande quantidade de tempo livre – isso aumenta a nossa necessidade para o entretenimento. Por consequência, a indústria de lazer também está a prosperar.

A sociedade moderna tornou-se tão complexa que ninguém consegue ter uma perspectiva geral e várias pessoas não sabem qual o papel que o seu trabalho desempenha no esquema global das coisas. Esta "alienação" é um grande problema social que está fora do âmbito deste livro. Mas mesmo nas grandes organizações existe um grau elevado de especialização de trabalho, que resulta na perda da perspectiva global por parte dos empregados que nem sempre se apercebem porque têm de fazer o que lhes foi pedido. Tal alienação pelo trabalho resulta num efeito negativo na produtividade. É por isso que várias organizações estão a organizar o seu trabalho de tal forma que os seus funcionários percebam claramente que estão a trabalhar para um cliente em particular. Entre os objectivos de um trabalho orientado ao consumidor, está o crescimento da motivação e consequentemente a produtividade dos funcionários. O facto de termos mudado de uma economia orientada ao fornecimento, na qual os meios de produção eram limitados, para uma economia orientada à demanda na qual são os clientes que são em número limitado, tem servido para reforçar esta tendência. Esta mudança de perspectiva, dos meios de produção para o cliente é também conhecida como o "mudança do paradigma organizacional" (ver figura 1.1).

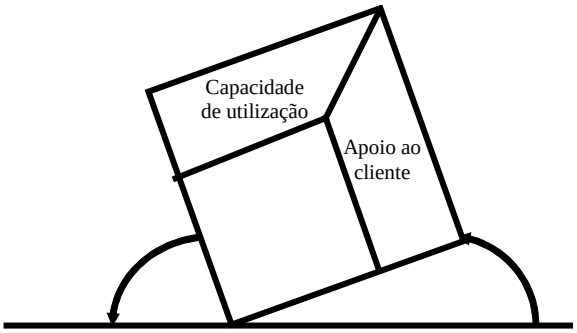


Figura 1.1. Mudança do paradigma organizacional

Os *sistemas de gestão de workflow* (*workflow management systems*) têm-se desenvolvido de forma a tornar o trabalho "controlável" e para encorajar a comunicação entre os funcionários. Esta é uma nova categoria de sistema de informação. Estes sistemas tornam possível construir, de uma forma linear, uma "ponte" entre o trabalho das pessoas e as aplicações computacionais.

1.3 Processos de Negócio

Existem vários tipos diferentes de trabalho, tais como, cozer pão, fazer a cama, desenhar uma casa ou recolher inquéritos com fins estatísticos. Em todos estes exemplos, podemos identificar a única "coisa" tangível que é produzida ou modificada: o pão, a cama, a casa ou a estatística. Neste livro, essa "coisa" será chamada de *caso*. Outros termos utilizados são: trabalho, tarefa, produto, serviço ou item. Um caso não tem de ser um objecto específico; pode ser mais abstracto – como um processo jurídico ou uma reclamação de um seguro. Um projecto de construção ou a montagem de carros numa fábrica são também exemplos de casos.

O trabalho sobre um caso é por natureza discreto. Isto é, cada caso tem um início e um fim, e cada um pode ser distinguido de todos os outros casos. Cada caso envolve um *processo* a ser realizado. Um processo consiste num número de *tarefas* que têm de ser executadas e um conjunto de *condições* que determinam a ordem das tarefas. Um processo pode também ser chamado de *procedimento*. Uma tarefa é uma unidade lógica de trabalho que é executada como um elemento indivisível por um recurso. Um recurso é um nome genérico para uma pessoa, máquina ou um grupo de pessoas ou máquinas que realizam tarefas específicas. Isto nem sempre significa que os recursos preocupam-se necessariamente em executar as tarefas independentemente, mas sim que são responsáveis por elas. Analisaremos mais profundamente este assunto na próxima secção.

Como exemplo de um processo, iremos examinar a forma como uma companhia de seguros (fictícia) trata de uma reclamação. Podemos identificar as seguintes tarefas:

1. *registar* a recepção da reclamação;
2. estabelecer o *tipo* de reclamação (por exemplo, fogo, carro, viagem, profissional);
3. verificar a *apólice* do cliente para confirmar que aquilo que ele está a reclamar é coberto pelo contracto que foi celebrado;
4. verificar o *prémio de seguro* para confirmar que os pagamentos estão em dia;
5. *rejeitar* se a tarefa 3 e 4 tiverem um resultado negativo;
6. escrever uma carta de *rejeição*;
7. estimar a *quantia a ser paga*, com base nos detalhes da reclamação;
8. nomear um *assessor* para averiguar as circunstâncias dos estragos e estabelecer o seu valor;
9. considerar *medidas de emergência* para limitar os estragos ou aliviar o problema;
10. estipular as *medidas de emergência* se for aprovado na tarefa 8;
11. estabelecer ou rever a *quantia a ser paga* e oferecida ao cliente;

12. anotar a *reacção do cliente*: aceitação ou discordância;
13. avaliação da *discordância* e decisão se deve rever (tarefa 11) ou tomar procedimentos legais (tarefa 14);
14. procedimentos legais;
15. *pagamento* da reclamação; e
16. *encerramento* da reclamação: arquivo.

Neste exemplo, podemos ver dezasseis tarefas que não precisam de ser realizadas necessariamente na ordem indicada. Duas ou mais tarefas que precisam de ser realizadas numa ordem estrita são designadas de *sequência*. Nalguns casos, algumas tarefas não precisam de ser executadas. Um exemplo é a nomeação de um perito, se o registo da reclamação está explícito e a quantia correspondente à reclamação é inferior a um determinado valor, o envolvimento de um perito não é necessário. Outras tarefas que nem sempre precisam de ser realizadas incluem tomar medidas de emergência, avaliar a discordância, ou realizar procedimentos legais. Portanto, às vezes temos de escolher entre duas ou mais tarefas. Chamaremos a isso *selecção*.

Existem também tarefas que devem ser realizadas em *paralelo*, como por exemplo, verificar a apólice e verificar os prémios do seguro. Estas são tarefas que têm obrigatoriamente de ser realizadas antes da tarefa "rejeição" começar. Designamos esta situação de *sincronização*.

Este exemplo de processo inclui também o conceito de *iteração*, ou repetição – isto é, a avaliação contínua de uma discordância ou a revisão da quantia a pagar. Em teoria, estes passos podem ser executados continuamente. A figura 1.2 mostra a ordem das tarefas usando um *diagrama de processos*: uma seta da tarefa A para a tarefa B significa que A tem de ser realizada antes de B.

Podemos também ver que o diagrama contém mais informação do que a lista de tarefas. Por exemplo, mostra-nos que uma determinada reclamação pode apenas ser terminada assim que algumas das medidas de emergência requeridas sejam tomadas. Cada tarefa está representada através de um rectângulo. Se uma tarefa tem mais do que uma tarefa sucessora – isto é, se tem mais de uma seta a sair – então precisamente *uma* das tarefas subsequentes tem de ser escolhida no decorrer da tarefa em questão. Se uma tarefa tem mais de uma tarefa antecedente – mais do que uma seta a incidir sobre ela – então *todas* elas têm de ser concluídas antes desta tarefa começar (sincronização). Os círculos indicam locais onde *workflows* específicos se encontram ou se dividem. Os círculos brancos têm *várias* tarefas antecedentes e apenas *uma* subsequente. Indicam que apenas uma das tarefas antecedentes precisa de ser realizada para continuar. Os círculos pretos têm *uma* tarefa antecessora e *várias* subsequentes. Mostram que todas as tarefas que se seguem têm de ser realizadas. (Os círculos podem ser observados como "tarefas fictícias"). O capítulo 2 introduz a notação de processos que torna mais fácil exprimir tais propriedades.

Resumindo, podemos identificar quatro mecanismos básicos distintos na estrutura de um processo: sequência, selecção, paralelismo e iteração. Na prática, todos eles são comuns e em princípio todos os processos podem ser modulados usando estes quatro elementos. Veremos estes elementos com mais detalhe no capítulo 2.

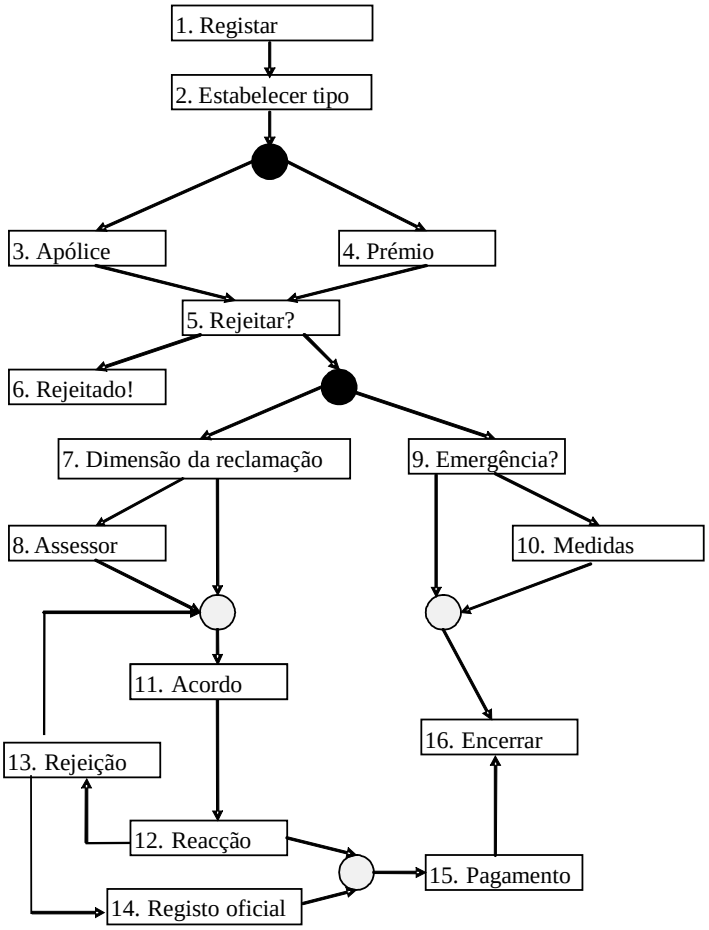


Figura 1.2. Processo de reclamação de seguro

Algumas tarefas podem ser realizadas pelo computador sem intervenção humana. Outras tarefas requerem inteligência humana: um julgamento ou uma decisão. Por exemplo, um empregado de um banco decide se será ou não concedido um pedido de empréstimo a um cliente. Trabalhadores humanos precisam de *conhecimento* para executar tarefas. Esse conhecimento está armazenado nas suas mentes por experiência, também conhecido por *conhecimento tácito*. Outras formas de conhecimento podem ser obtidas pela aprendizagem e recolha de informação, também chamado de *conhecimento explícito*. A *gestão de conhecimento* está relacionada com a aquisição, enriquecimento e distribuição de conhecimento, para que o conhecimento adequado esteja disponível no momento exacto à pessoa que irá realizar uma tarefa.

Uma tarefa pode também ser definida como um processo que não pode ser subdividido: um processo *atómico*. No entanto, existe um elemento subjectivo nesta definição – o que uma pessoa considera uma tarefa única poderá ser não atómica para outra pessoa. Por exemplo, para uma companhia de seguros, a

elaboração do relatório aos danos de um carro é uma tarefa única para o assessor, no entanto, para um especialista trata-se de um processo que compreende várias tarefas, como verificar o chassis, motor e carroçaria. Portanto, uma tarefa é um processo atómico para a pessoa que a define ou a ordena, mas para a pessoa que a executa, trata-se de muitas vezes uma tarefa não atómica.

Em cada caso é realizado um único processo. Chamamos ao desempenho de uma tarefa por um recurso, uma *actividade*. Vários casos podem ter o mesmo processo, mas cada caso pode seguir um percurso diferente nesse mesmo processo. Por exemplo, na companhia de seguros, uma reclamação pode envolver uma rejeição e outra não. O percurso depende das características específicas do caso – os *atributos do caso*. O número de processos numa organização é (geralmente) finito e mais pequeno que o número de casos a considerar. Como resultado, uma organização pode desenvolver uma rotina para desempenhar processos e assim fazer com que funcione eficientemente.

Isto pode ser visto claramente na indústria têxtil: é muito mais rápido fazer cem saias com o mesmo padrão do que cem saias usando padrões diferentes. Produtos uniformizados são mais baratos do que se forem feitos à medida. Produzir mil saias com o mesmo padrão sai menos dispendioso do que fazer dez vezes cem saias com o mesmo padrão. Isto é designado de economia de escala: os custos por caso descem quando o número de casos aumenta. Portanto, as organizações esforçam-se por minimizar o número de processos e tentam aumentar ao máximo o número de casos que cada um pode desempenhar – pelo menos enquanto ganharem alguma coisa com cada caso. Afinal de contas, o lucro é o objectivo essencial.

Uma companhia de seguros deseja manter o número de reclamações o mais baixo possível – mas geralmente este é um factor que não se pode controlar. Tentam também reduzir ao mínimo o número de processos. Existe no entanto, uma contrapartida: os processos não se devem tornar muito complicados. É preferível ter mais alguns processos, que sejam mais simples, do que ter um número menor de processes que sejam extremamente complexos. Lembremo-nos de que, na teoria, é possível combinar dois ou mais processos em apenas um, como está ilustrado na figura 1.3. Os processos A e B juntam-se para formar um só processo, o processo C.

Aqui foi acrescentada uma tarefa adicional: decidir qual o tipo de caso com que estamos a lidar e escolher um dos processos para continuar. Isto é portanto um falso ganho. De forma a alcançar uma estrutura de processo eficiente, é necessário realizar cálculos que não se podem executar sem o auxílio de simulações computacionais.

A situação que descrevemos anteriormente é a mais comum: um pequeno número de processos com um grande número de casos. No entanto, existem excepções à regra. Por exemplo, cada fato que um alfaiate produz é feito à medida; podemos então dizer que ele tem de desenhar e recomeçar um novo processo para cada caso. Isto aplica-se também a um arquitecto que tem de esboçar uma nova casa ou conjunto de escritórios a partir do zero. Podemos ver isto ainda de outra forma: tanto o alfaiate como o arquitecto utilizam certamente uma abordagem *standard*, e, por consequência, um processo que seguem sempre. O alfaiate começa por tirar as medidas do cliente, depois mostra-lhe um número de padrões e tenta determinar com ele qual deles melhor se adequa ao seu gosto, então realiza as primeiras mudanças ao padrão. Após ter sido escolhido o tecido, o alfaiate começa a desenhar o modelo. Existem também

muitas outras tarefas que podem ser identificadas como parte de cada caso. O mesmo se aplica ao arquitecto. O que podemos ver neste cenário é que existe efectivamente um processo, mas que as tarefas desempenhadas são dependentes de um caso. Por isso, existe uma medida para a complexidade de um processo: o grau de dependência das tarefas aos casos.

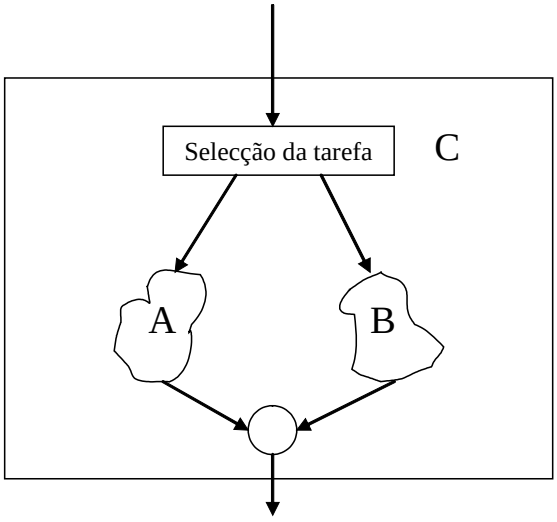


Figura 1.3. Junção de dois processos em apenas um

Embora devamos primeiramente lidar com as situações em que temos muitos casos para um processo, há muitas situações em que é necessário desenhar um novo processo para cada caso. A estes designamos de "processos únicos". Nestes, o primeiro estágio para representar o caso é o desenho do seu processo específico. Mesmo nesta situação, frequentemente o processo é composto por tarefas *standards*. Nestas situações dizemos que cada caso tem o seu próprio *projecto*. Aqui as palavras "projecto" e "processo" são sinónimas.

Vimos anteriormente que o trabalho realizado em cada caso é de natureza discreta: cada um tem um único começo e um fim. No entanto, existe também trabalho de natureza contínua que não pertence certamente a um único caso. Vejamos, por exemplo, o porteiro cujo trabalho consiste em dar assistência às pessoas que entram num edifício, ou um polícia que garante a segurança de um distrito fazendo patrulha. Em ambos os exemplos, um caso pode ainda – com um pouco de boa vontade – ser definido, identificando períodos e considerando guardar a porta ou efectuar uma patrulha por um período específico como um caso. Assim, o empregado recebe automaticamente uma sequência contínua de casos, um para cada período. Outra forma de tratar o trabalho de natureza contínua em termos de casos, é considerar o trabalho como um todo como um caso que compreende uma repetição contínua das tarefas. Neste livro concentramo-nos no trabalho discreto – mas não excluímos o trabalho contínuo. Pode servir como um exemplo extremo no qual podemos testar os princípios apresentados neste livro.

Para concluir esta secção, vamos subdividir os processos em três tipos: *primário*, *secundário* e *terciário*:

- Os processos primários são aqueles que produzem os produtos e os serviços de uma empresa. Consequentemente, são também conhecidos por *processos de produção*. Lidam com os casos relacionados com os clientes. Como regra, são processos que geram as receitas da empresa e são claramente orientados ao cliente. Por vezes, o cliente ainda não é conhecido quando as empresas produzem para armazenamento. Alguns exemplos de processos primários são a aquisição de materiais e componentes, a venda de produtos e serviços, o desenho e a engenharia, e a produção e distribuição.
- Os processos secundários são aqueles que suportam os primários. Consequentemente, são chamados de *processos de suporte*. Um grupo importante de processos secundários concentra-se em manter os meios de produção: a aquisição e manutenção da maquinaria, os veículos e as instalações. Um grupo comparável de processos é o que envolve a gestão de pessoal: recrutamento e selecção, formação, apreciação de trabalho, salários e despedimentos. A administração financeira é também um processo secundário, tal como o *marketing*.
- Os processos terciários são os *processos de gestão* que direccionam e coordenam os processos primários e secundários. Nestes processos, os objectivos e pré-condições sobre os quais os gestores de outros processos operam, são formulados e os recursos requeridos para a execução dos outros processos são reservados. Os processos de gestão também abrangem a manutenção dos contactos com os financiadores e outros *stakeholders*.

A figura 1.4 mostra as relações entre os três tipos de processos.

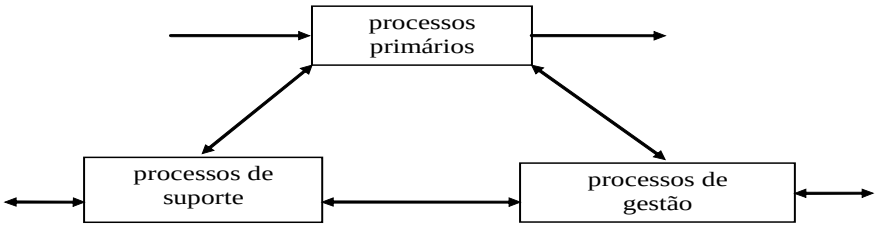


Figura 1.4. Ligações entre os três tipos de processos

Os processos de gestão têm objectivos e capital como entrada e devem atingir um desempenho satisfatório – muitas vezes na forma de lucro. Os processos de suporte recebem dos processos de gestão os meios para comprar recursos e libertam os recursos que já não precisam. Os recursos geridos pelos processos secundários são postos à disposição dos processos primários que os devolvem depois de os usar. Como entrada, os processos primários recebem, por um lado, ordens, e, por outro lado, componentes e matéria-prima. Como saída, entregam produtos e serviços. Recebem tarefas e orçamentos de compra dos processos de gestão. Os processos primários e de suporte reportam aos processos de gestão e submetem os lucros.

Os processos secundários e terciários são muitas vezes de natureza contínua embora possam conter sub-processos discretos, por outro lado, os processos primários são geralmente conduzidos por casos tendo portanto um carácter discreto.

1.4 Distribuir e Aceitar Trabalho

Os animais e as máquinas trabalham em resposta a *ordens* ou *atribuições* feitas por pessoas. Mas, a maior parte do trabalho das pessoas é-lhes também atribuído por outras pessoas: os seus supervisores. Os artistas, os cientistas ou os políticos são excepções que podem – dentro de certos limites – decidir por si próprios que trabalho irão realizar.

Existem dois tipos de supervisores: o *patrão* e o *cliente*. Em última instância, as tarefas atribuídas pelos patrões estão directa ou indirectamente relacionadas com o trabalho efectuado para os clientes. Esse relacionamento é "directo" se o trabalho realizado traz resultados sob a forma de um produto ou serviço para um cliente, que pode até ser desconhecido. Geralmente isto aplica-se aos processos primários. Por outro lado, o relacionamento é "indirecto" se o trabalho envolve a manutenção ou o melhoramento dos processos de produção: os processos secundários e terciários.

Na maioria das organizações existe uma hierarquia na qual as tarefas que foram atribuídas a determinadas pessoas podem, em parte, ser passadas para outras pessoas num nível inferior dessa mesma hierarquia. Uma pessoa à qual é atribuída uma tarefa é um *fornecedor*, também designado por *recurso*. Costumamos utilizar este último termo pois as tarefas podem ser realizadas por máquinas – em particular, aplicações informáticas – assim como por pessoas. Temos até agora tratado os supervisores e fornecedores como pessoas individuais, mas esses papéis podem ser desempenhados por departamentos de uma dada empresa ou por firmas individuais. Iremos consequentemente utilizar o termo *actor* para descrever supervisores e os fornecedores em geral. Um actor pode desempenhar ambos os papéis – supervisor ou fornecedor (ou recurso) – ao mesmo tempo.

Um fornecedor não tem necessariamente de realizar o trabalho por si próprio, mas pode redireccionar ou subcontratar a terceiros. Mas, em qualquer dos casos, o fornecedor tem sempre de *dirigir* o trabalho que aceita.

Em organizações de maiores dimensões, os funcionários cumprem as ordens recebidas sem muitas vezes saberem para que cliente uma tarefa está a ser realizada. Um caso particular é aquele em que os produtos estão a ser produzidos para serem armazenados em *stock*, uma vez que durante a produção o cliente é ainda desconhecido. (E às vezes não existe realmente um cliente para um determinado produto).

Como foi referido anteriormente, um supervisor pode ser tanto um patrão como um cliente. Existe também uma grande variedade entre os clientes. Para o Serviço Prisional, os criminosos (prisioneiros) são os clientes; os clientes do Fisco são os contribuintes enquanto que para os hospitais são os pacientes. O papel de um cliente depende da situação: o padeiro é o cliente do jardineiro quando o jardineiro trata do jardim deste. Por outro lado, o jardineiro é o cliente do padeiro quando vai ter com ele para lhe comprar pão.

Em grandes organizações, há uma forte tendência para acentuar mais claramente o papel do cliente. O princípio de que "o cliente tem sempre razão" está a ganhar terreno em relação à idea "trabalhar para o patrão". O espírito informado do cliente garante que os funcionários estejam mais conscientes do destinatário do seu trabalho, o que leva a uma maior atenção ao seu trabalho: afinal de contas, se entregarem um trabalho de fraca qualidade, irão ficar incertos de que o cliente queira ou não o seu trabalho. (Para um "cliente" de uma prisão, este princípio funciona de forma contrária).

Para qualquer trabalho, existe um supervisor e um fornecedor que têm – por vezes não escrito – um *contrato* entre si sobre o caso a ser realizado, o prazo final para concluir o projecto e o preço a pagar. Se o fornecedor é uma empresa externa, então os processos de comunicação serão criados entre o supervisor e o fornecedor antes do contracto ser estabelecido, e as comunicações entre os dois actores poderão continuar a ser necessárias durante a realização da tarefa. Quando a relação entre o fornecedor e o supervisor é formalizada, podemos verificar a existência de um *protocolo de comunicações*. No entanto, isto pode ser muito complexo. A figura 1.5 mostra um exemplo de um protocolo de comunicação.

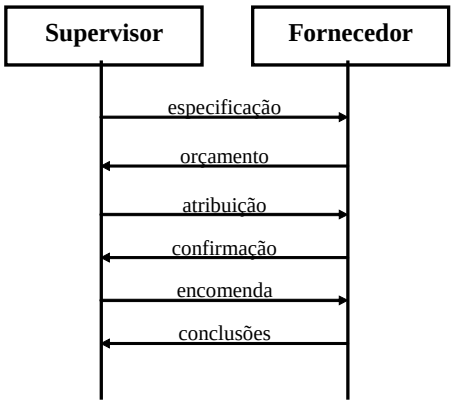


Figura 1.5. Protocolo de comunicação

Neste exemplo, podemos ver os passos executados no relacionamento entre as duas entidades envolvidas. Primeiro, o supervisor fornece a especificação do trabalho a ser realizado. Depois, o fornecedor elabora um plano para realizar o trabalho e estabelece um preço, isto é o "orçamento" que é submetido ao supervisor. O supervisor recebe o orçamento e encomenda o trabalho. Na prática, poderá haver uma discussão intensa entre as várias partes durante um certo período de tempo, com o supervisor a fazer pedidos suplementares – por exemplo, acerca do preço – e o fornecedor a explicar como pensa realizar o trabalho. Em muitos casos, o momento em que uma ordem é confirmada não é o mesmo que o início do trabalho. Se o trabalho faz parte de um projecto mais amplo que o supervisor está a gerir, então este apenas terá início assim que os outros elementos do projecto tenham sido concluídos: assim, o supervisor determina em que ponto o trabalho pode começar. O número de passos que existem num protocolo de comunicação entre um supervisor e o fornecedor

pode variar de caso para caso, de acordo com as características específicas e o tratamento de cada um e não necessita de estar predeterminado.

Um actor responsável por um processo pode atribuir uma tarefa como um todo a um fornecedor ou pode decompô-la num processo, isto é, uma rede de tarefas, cada uma das quais atribuída a um fornecedor. Por sua vez, estes fornecedores podem repetir este processo de decomposição. Esta decomposição produz uma *árvore de contractos*. A execução de uma tarefa para um caso particular requer a execução de um protocolo de comunicação entre supervisores e fornecedores. Em vez de decompor uma tarefa num processo e fazer o *outsourcing* das subtarefas para todos os casos que passam por uma subtarefa, é possível fazer isto de maneira diferente para cada caso. Assim, a execução de uma tarefa para um caso específico começa com a "fase de desenho", na qual é criada uma rede de tarefas em que os (sub)fornecedores são seleccionados. A figura 1.6 mostra um exemplo disto. Neste exemplo, a tarefa é o transporte de carga do ponto A para o ponto K. O supervisor P subdivide este trabalho em duas tarefas: transporte do ponto A para o ponto D e transporte do ponto D para o ponto K. Cada uma destas tarefas é subcontratada a um fornecedor diferente, que são o fornecedor Q e R. Cada uma das tarefas é de novo subdividida por estes dois: pelo supervisor/fornecedor Q no transporte de A para C e então depois de C para D, e pelo supervisor/fornecedor R de D para J e seguidamente de J para K. Podemos ver a ilustração deste exemplo na figura 1.6. Note que tanto Q como R agem como supervisor *e* fornecedor.

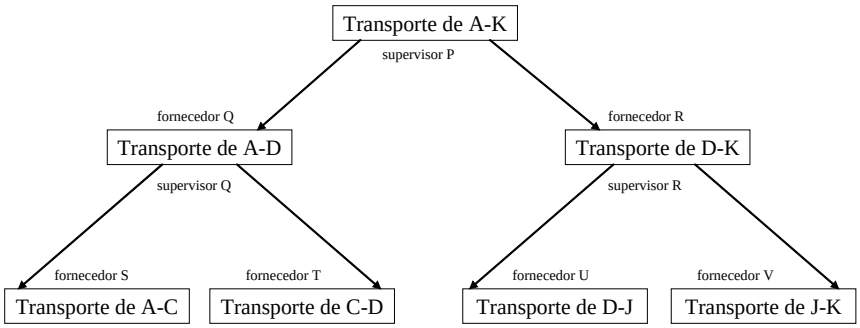


Figura 1.6. Árvore de contratos

Esta árvore contém "nós", que são indicados no exemplo através de rectângulos. Os "ramos" ligam dois "nós". Os "nós" mostram os actores que estão responsáveis por uma parte do trabalho. Neste exemplo, os actores são identificados pelas tarefas que têm de desempenhar. A "raiz" da árvore (que se encontra no topo do diagrama) recebe as atribuições de tarefas directamente do seu supervisor. As "folhas" da árvore (isto é, as ramificações mais baixas dos "nós") são os actores que realizam as tarefas. Os outros actores são simultaneamente supervisores e fornecedores. Um actor X é um subfornecedor de outro actor Y se existir um arco de Y para X. Um actor é um supervisor se existir um arco que o conduz a outro actor. Consideremos o exemplo da figura 1.6. O actor Q é um subfornecedor de P e o supervisor de S e T. Estes processos de decomposição e *outsourcing* ocorrem frequentemente dentro da própria organização mas também entre diferentes organizações. Nos negócios

electrónicos tentamos automatizar/computorizar estes processos tanto quanto possível. Se quisermos suportar os processos de negócio através dos sistemas de informação, precisamos de descrições muito detalhadas e precisas destes processos de negócio. Se quisermos juntar processos de negócio de organizações diferentes de forma automática/computorizada, isto torna-se ainda mais importante.

1.5 Estruturas Organizacionais

Uma grande quantidade de literatura foi publicada sobre as estruturas organizacionais e qualquer tentativa de resumi-la em poucos parágrafos está destinada a falhar. Por consequência, esse não será o nosso objectivo. Porém, discutiremos as propriedades das três formas mais importantes de uma estrutura organizacional que são relevantes ao desenvolvimento de *workflows*.

Uma estrutura organizacional estabelece de que forma o trabalho realizado pela organização é dividido pelos seus funcionários. Na maioria dos casos, a divisão não é feita pelas próprias pessoas, mas antes pelos *papéis* e pelas *funções* que elas desempenham. Cada pessoa pode cumprir diversos papéis durante a sua vida. Algumas podem, por exemplo, começar como assistente administrativo e terminar como chefe de contabilidade. As pessoas podem também realizar vários papéis ao mesmo tempo. Pode ser que a mesma pessoa seja, tanto um motorista como um estafeta que entrega mensagens quando não existe ninguém para ser conduzido. Um aspecto importante de uma estrutura organizacional é a divisão da autoridade e da responsabilidade. Se um executivo tem responsabilidades específicas, então também tem de ter autoridades particulares. Estas envolvem frequentemente a autoridade de atribuir trabalho a outros membros da equipa de funcionários, ou seja, subcontratar trabalho a outros. Por outro lado, um executivo é também responsável por assegurar que o trabalho que lhe é atribuído, por colegas autorizados, se realize.

As três formas mais importantes de estruturas organizacionais, ou melhor, de mecanismos de coordenação, são:

1. a organização hierárquica;
2. a organização em matriz; e
3. a organização em rede.

Destes três mecanismos, a *organização hierárquica* é a mais conhecida, sendo caracterizada por uma estrutura em árvore. Tal estrutura é designada de *gráfico organizacional*. Na secção anterior já tínhamos mencionado estruturas em árvore na forma de árvores de contrato. Num gráfico organizacional, cada "nó" que não seja uma "folha", indica um papel ou uma função individual. Normalmente, as "folhas" da árvore representam grupos de funcionários ou departamentos. Os "ramos" mostram relações de autoridade: a pessoa no início (topo) de um "ramo" está autorizada a requisitar trabalho à pessoa ou departamento localizada no nível inferior.

Também existe uma outra definição para o gráfico organizacional, que se parece muito com a nossa mas é, de facto, diferente. Nesta definição, cada "folha" representa uma pessoa e cada nó num nível mais elevado representa um departamento. O nó da "raiz" indica a organização inteira e todos os outros nós

uma parte dela. As pessoas indicadas em cada folha pertencem assim ao departamento representado no nó imediatamente acima. Enquanto que a primeira definição mostra uma pessoa que é responsável por todas aquelas que estão abaixo dela na árvore para quem essa pessoa representa a raiz, a segunda definição olha para cada um desses conjuntos de funcionários como um departamento. A semelhança entre os gráficos organizacionais e as árvores de contrato é que ambos expressam relações do tipo supervisor–fornecedor usando "ramos". A diferença é que num gráfico organizacional esta relação não está ligada a nenhum caso específico, enquanto que, esta relação é muito importante para uma árvore de contratos. Numa organização estritamente hierárquica, a comunicação entre dois "nós" passa sempre através do antecessor comum mais próximo de ambos. A figura 1.7 mostra um exemplo de um gráfico organizacional.

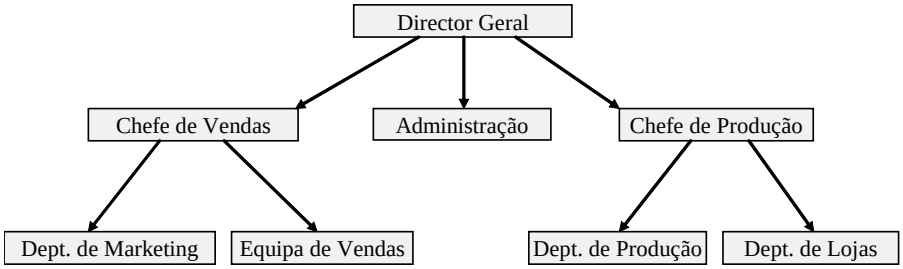


Figura 1.7. Gráfico organizacional

Neste exemplo, a comunicação formal entre a equipa de vendas e o departamento de lojas é feita através do chefe de vendas, do director geral e do chefe de produção. A "gestão" ou a "administração" é frequentemente a "raiz" de um gráfico organizacional. As suas "folhas" são os departamentos da organização. Um exemplo típico de uma organização hierárquica é o exército. Na prática, existem muitas conversas informais entre os funcionários e membros de departamentos, permitindo que a comunicação seja mais rápida do que se seguíssemos as linhas hierárquicas. As organizações puramente hierárquicas estão agora praticamente extintas, pois esta estrutura é muito inflexível. Em muitas empresas é demasiado pesado permitir a delegação de trabalho somente através de canais fixos hierárquicos.

No desenho de uma organização hierárquica, somos livres de escolher que departamentos são criados e que camadas de gestão existem acima deles. Na distribuição dos funcionários pelos departamentos, podemos seleccionar entre três princípios:

- *O grupo de capacidade* – Pessoas com as mesmas competências são colocadas juntas no mesmo departamento. Em princípio, tais pessoas podem ser substituídas por colegas do mesmo grupo. A tarefa do presidente do departamento é manter os seus membros "actualizados" – através de formação, por exemplo – e fazer todo o possível para "publicita-los" a outras unidades de negócio para as quais desempenham trabalho. Exemplos típicos incluem conjuntos de dactilógrafos e equipas de engenheiros de manutenção.

- *O departamento funcional* – Este executa um grupo interdependente de tarefas, cada uma requer frequentemente as mesmas competências. A responsabilidade pelo trabalho do departamento depende do seu chefe. Exemplos típicos incluem os departamentos de contabilidade, de marketing e de manutenção.
- *Departamentos de processos ou de produção* – Neste caso, o departamento é responsável por um processo de negócio completo ou pelo fabrico de um produto.

O primeiro ou o segundo tipo de organização é escolhido frequentemente para os processos secundários. Nos processos primários, a terceira forma de organização começa a ganhar importância. As camadas de gestão hierárquicas encontram-se sobre os departamentos. Na escolha destes, a seguinte questão tem um papel importante: será elevado ou reduzido o nível de coordenação necessário entre os departamentos? Deve existir um número mínimo de camadas entre os departamentos que precisam de estabelecer uma coordenação intensa entre si, devendo ter preferencialmente um único gestor.

Um gestor tem uma *abrangência de controlo* máxima. Ou seja ele não pode administrar um número ilimitado de subordinados. Tal abrangência dependente em grande parte da natureza do trabalho e da própria experiência do gestor.

Foi desta forma que a *organização em matriz* surgiu. Esta forma de organização é estruturada de acordo com duas dimensões: a *funcional* e a *hierárquica*. A parte hierárquica é a mesma que foi descrita anteriormente e é geralmente baseada em grupos funcionais ou de capacidade: as pessoas com as mesmas competências pertencem ao mesmo grupo. A parte funcional é baseada nas tarefas que têm de ser executadas. (A terminologia pode ser confusa). Cada pessoa tem assim um patrão hierárquico – o chefe do departamento ao qual ela pertence – e um patrão funcional, que é responsável pela tarefa a ser realizada. As tarefas – que no contexto de organizações em matriz são geralmente chamadas de "projectos" – são únicas; por outras palavras, nenhuma estrutura fixa pode ser criada com base nas tarefas, por isso a estrutura hierárquica (fixa) baseia-se nas competências das pessoas. Os patrões funcionais são conhecidos como "líderes de projectos".

Organizações em matriz são encontradas sobretudo em organizações que operam na base de projectos, tal como construtores, empresas de instalação e empresas de desenvolvimento de *software*, ou seja, em negócios que não realizam produção em série, mas sim projectos únicos. A estrutura funcional está assim constantemente sujeita à mudança. É muito provável que a pessoa A seja durante algum tempo líder de um projecto no qual a pessoa B participa, e B posteriormente fique líder de um projecto que envolva a pessoa A. A figura 1.8 mostra um exemplo da distribuição de uma equipa de funcionários numa organização em matriz. As colunas mostram a distribuição funcional e as linhas a distribuição hierárquica.

Podemos ver como uma pessoa pode fazer parte de um ou mais projectos. Naturalmente, uma pessoa pode estar envolvida apenas num projecto, num dado período de tempo, mas é igualmente possível para alguém trabalhar alternadamente em diversos projectos durante o mesmo período de tempo. Frequentemente várias pessoas de um departamento trabalham num mesmo projecto. Na matriz isto significaria ter mais do que uma pessoa incluída na mesma célula. Por uma questão de simplicidade tal não foi representado na

figura 1.8. Uma forma de organização que se assemelha fortemente à organização em matriz ocorre quando os processos são geridos por um gestor de processos e os casos por um *gestor de casos*. O primeiro é responsável pela qualidade e eficiência do "seu" processo, enquanto que o segundo assegura a conclusão rápida e eficiente dos "seus" casos. Isto pode conduzir a um conflito de interesses.

	Projecto-1	Projecto-2	Projecto-3
Supervisores	Luísa	Anita	João
Carpinteiros	Pedro	Karl	Geraldine
Pedreiros	Henrique	Tom	Jerry
Pintores	Berta	Simone	Simone
Estucadores	Carlos	Pedro	Paulo

Figura 1.8. Distribuição dos funcionários numa organização em matriz

A última forma de organização que podemos identificar é a organização em rede. Nesta, actores autónomos colaboram para fornecer produtos ou serviços. Para o cliente, eles parecem ser apenas uma organização, razão pela qual uma organização em rede é, por vezes, denominada de *organização virtual*. Os actores actuam como supervisores e fornecedores. A autonomia implica que não existe nenhuma relação permanente formal (de emprego), o que significa que um actor pode escolher se deseja ou não realizar uma dada tarefa. Os actores necessários para executar cada tarefa devem consequentemente ser recrutados individualmente em cada ocasião. Isto pode ser feito através de um protocolo e de uma árvore de contrato, como foi discutido na secção anterior. Isto pode consumir bastante tempo, assim estruturas de contratos são uma alternativa frequentemente escolhida para contratações regulares. Tal contrato determina que um colaborador está disponível para executar um determinado tipo de trabalho. Tal como na organização em matriz, o actor A pode ser director do actor B para um tipo de trabalho mas subcontratado para outro.

Cada vez mais as organizações em rede estão proliferando. Existem duas razões principais que contribuem para isso. Primeiro, as empresas estão a tentar manter a sua força de trabalho permanente tão pequena quanto possível fazendo uma utilização extensiva de trabalho temporário e de subcontratados. Isto, juntamente com o facto de muitas pessoas actualmente trabalharem em *part time*, é conhecido por flexibilização do trabalho. Deste modo as empresas podem controlar os seus custos fixos. O uso de *co-produtores* e de *outsourcers*, que são exemplos de fornecedores, é muito comum nas indústrias da construção e de motores. A segunda razão é que as organizações especialistas, cada uma com uma gama limitada de produtos, podem fornecer em conjunto um produto completo. Exemplos podem ser encontrados na indústria da construção, no qual um leque de actores juntam forças para construir uma ponte, e empresas de consultoria, que agregam conhecimento individual para oferecer um produto integrado que incorpora, por exemplo, conhecimentos financeiros, legais, fiscais e apoio na área das TI (Tecnologias de Informação). Uma organização em rede é, de certa forma, comparável com uma organização em matriz. Apesar de tudo,

os recursos para cada projecto são configurados individualmente. A diferença é que neste caso os recursos não têm o mesmo empregador.

1.6 Gestão de Processos

Uma forma estabelecida de estudar a gestão de processos é distinguir entre um *sistema de gestão* e um *sistema gerido*. A palavra "sistema" significa aqui todas aquelas pessoas, máquinas e sistemas de informação computorizados que realizam processos específicos. Um sistema gerido pode ser subdividido num sistema de gestão e num sistema gerido de nível mais baixo (ver figura 1.9). O sistema gerido no mais baixo nível desta subdivisão é um *sistema de execução*. No nível mais elevado, um sistema é sempre parte de um sistema gerido. Um sistema de gestão pode gerir vários sistemas, assegurando a capacidade dos sistemas geridos para comunicarem uns com os outros e com o mundo exterior – isto é, o sistema gerido num nível superior.

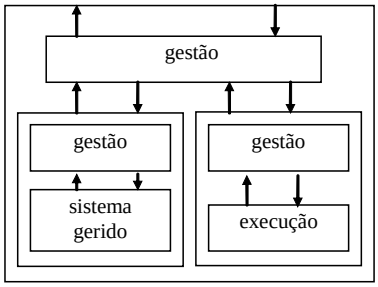


Figura 1.9. Paradigma de gestão recursivo: toda a entidade é um sistema gerido

Entre o sistema de gestão e o sistema gerido ocorre uma troca de informação. Isto permite ao sistema de gestão comunicar objectivos, pré-condições e decisões ao sistema gerido, e este, por outro lado, informa o sistema de gestão. Com base nessa informação, o sistema de gestão pode rever os objectivos, as pré-condições e as decisões. Estas fases são geralmente denominadas de *ciclo de controlo e planeamento* que pode ser identificado em todas as organizações.

A gestão de processos tem sido desde há muito tempo dividida em quatro níveis. A diferença entre estes está baseada na frequência e na abrangência das decisões a serem tomadas. Por abrangência referimo-nos a dois factos: o período de tempo durante o qual a decisão tem influência e o seu (potencial) impacto financeiro. Os quatro níveis são os seguintes (ver figura 1.10):

1. *Gestão em tempo real* – As decisões podem ser tomadas frequentemente (num intervalo de alguns micro segundos até horas). O período de tempo durante o qual a decisão tem um efeito é muito curto e as consequências financeiras de uma decisão errada são pequenas.

2. *Gestão operacional* – As decisões são tomadas regularmente (de horas a dias) e a sua abrangência é limitada. Ou seja, a influência da decisão não é perceptível depois de um curto período de tempo.

3. *Gestão táctica* – As decisões são tomadas periodicamente (de dias a meses) e a sua abrangência é limitada.

4. *Gestão estratégica* – As decisões são tomadas apenas uma vez, ou não mais do que cada dois anos, e a sua abrangência é maior. A influência de uma decisão estratégica pode permanecer por muitos anos.

Nível de gestão	Tempo	Impacto financeiro	Tipo de decisões	Métodos suportados
Tempo real	Segundos – horas	Baixo	Controlo de equipamento	Teoria de controlo
Operacional	Horas – dias	Limitado	Atribuição de recursos	Optimização combinatória (ex: agendamento)
Táctico	Dias – meses	Elevado	Capacidade de planeamento de recursos e orçamentação	Modelos estocásticos (e.g., modelos de filas)
Estratégica	Meses – anos	Muito elevado	Desenho de processos e tipos de recursos	Modelos financeiros, analise multi-critério

Figura 1.10. Os quatro níveis de gestão de processos

Uma outra diferença entre estes níveis de gestão é o tipo de decisões que são tomadas. A gestão em tempo real e a gestão operacional envolvem somente

aspectos dinâmicos, não a estrutura dos processos de negócio. A gestão em tempo real envolve o controlo de máquinas e veículos. A gestão operacional trata principalmente da distribuição de recursos aos casos e o encaminhamento dos mesmos. Exemplos típicos de gestão operacional são o planeamento da produção e encaminhamento de comboios.

Os objectivos da gestão táctica dizem respeito ao *planeamento de capacidades* e *orçamentação* para a gestão operacional. O planeamento de capacidade envolve determinar as quantidades de recursos requeridos conforme o tipo de caso. Isto não significa apenas recursos humanos, mas também as máquinas e matérias-primas usadas na execução de um caso. A gestão de *stock* é um exemplo típico, envolvendo não apenas a gestão dos próprios *stocks* de matérias-primas mas também a gestão dos recursos de reserva. A orçamentação preocupa-se com a obtenção de meios financeiros e com a formulação de objectivos em termos financeiros.

A gestão estratégica debruça-se sobre os aspectos *estruturais* dos processos e tipos de recursos. Uma questão estratégica coloca-se quando uma organização está indecisa entre executar ela própria um processo em particular, ou subcontrata-lo. Outra questão visa determinar de que forma os processos devem ser estruturados e que procedimentos devem ser seguidos.

Cada nível de gestão, à excepção da gestão em tempo real, tem também a tarefa de tratar das excepções às regras que ocorrem nos níveis inferiores. A gestão táctica pode ser envolvida se a alocação do recurso ao nível operacional não tiver sucesso.

Tomar decisões é uma característica importante da gestão (de processos). A disciplina de investigação operacional (IO) procura as melhores soluções possíveis para problemas de decisão, usando técnicas matemáticas. A inteligência artificial (IA) tenta desenvolver sistemas de computação que imitam as técnicas humanas para resolver problemas de decisão (heurísticas). A sociologia organizacional dedica a sua atenção aos métodos pelos quais as pessoas podem cooperar para encontrar uma solução. Nesta secção iremos concentrar-nos na elaboração de um resumo das quatro fases da resolução de problemas de decisão:

- *Definição* – Envolve estabelecer exactamente qual é o problema e, em particular, qual o domínio da solução. O estabelecimento de critérios de optimização faz normalmente parte desta fase.
- *Criação* – Envolve formular uma ou mais soluções que estejam dentro do domínio definido ou que satisfaçam um critério de optimização.
- *Avaliação* – Envolve avaliar soluções diferentes, por exemplo, pela análise multi-critério.
- *Selecção* – Envolve escolher uma solução que funcione com o intuito de implementá-la.

Em princípio, um suporte computacional está disponível para todas estas tarefas, particularmente para a segunda e terceira. Isto, por vezes, é possível usando uma simples folha de cálculo, mas geralmente requer técnicas matemáticas ou modelos de simulação.

1.7 Sistemas de Informação para Processos de Negócio

A organização do trabalho, tanto dentro como entre empresas, está a tornar-se cada vez mais complicada. E por isso, sistemas de informação (computorizados) foram desenvolvidos para suportar a gestão de processos e a sua coordenação. Primeiro iremos apresentar um método para classificar sistemas de informação. Depois esboçaremos como foram desenvolvidos no passado e como serão, provavelmente, desenvolvidos no futuro próximo.

Os sistemas de informação podem ser categorizados de várias formas. Aquela que escolhemos utilizar é baseada no papel desempenhado pelos sistemas nos processos. A lista seguinte está organizada por ordem ascendente de funcionalidade: o primeiro tipo de sistemas listado contém pouquíssimo conhecimento sobre processos e só deve ser usado para apoiar as pessoas que realmente fazem o trabalho, enquanto que o último tipo pode gerir processos sem qualquer intervenção humana:

- *Sistemas de informação de escritório*. Estes sistemas ajudam a equipa de funcionários responsável por realizar e gerir os processos com processamento de informação básica: escrita, desenho, cálculo, arquivo e comunicação. Este tipo de sistema inclui processadores de texto, ferramentas de desenho, folhas de cálculo, sistemas de gestão de bases de dados e correio electrónico. Estes sistemas não possuem qualquer conhecimento sobre processos. Apesar de a informação que processam possa conter conhecimento sobre o negócio, estes sistemas não podem utilizar esse conhecimento.
- *Sistemas de processamento de transacções*. Estes sistemas, também chamados de sistemas de registo, registam, comunicam e gravam os aspectos relevantes das mudanças dos processos. Os sistemas de processamento de transacções que se especializam na comunicação entre organizações diferentes são chamados de *sistemas de informação inter-organizacionais*. Estes usam frequentemente *Electronic Data Interchange* (EDI) usando *standards* para a troca de dados, como por exemplo, o XML. O "coração" de tal sistema é geralmente um sistema de gestão de base de dados, mas hoje em dia um sistema de gestão de *workflow* torna-se também um componente essencial. Este último tipo de sistema tem algum conhecimento dos processos, como comprovado – por exemplo – pelo facto de poder interpretar transacções e determinar onde e como os dados de entrada devem ser armazenados.
- *Sistemas de gestão de conhecimento*. São responsáveis pela aquisição e distribuição de conhecimento a ser usado por trabalhadores de conhecimento, trabalhadores ou gestores de um caso. O conhecimento que controlam é conhecimento explícito que pode ser representado de forma digital. Um motor de procura interligado a um sistema de gestão de documentos é um dos exemplos mais simples de um sistema de gestão de conhecimento. Com tal sistema, um trabalhador de conhecimento é capaz de encontrar fragmentos relevantes de texto produzidos por ele ou por outros trabalhadores, por meio da pesquisa de texto livre ou de palavras-chaves. Um sistema mais avançado é um sistema de raciocínio com base em casos (*case-based reasoning system*) que pesquisa uma base de dados de casos ideais para encontrar casos semelhantes a um caso inicialmente especificado.

As soluções apresentadas pelos casos encontrados podem ser aplicáveis ao caso real. Os gestores estão interessados, na maior parte das vezes, em dados agregados sobre o processamento dos casos ou sobre os próprios casos. Por isso, usamos muitas vezes *data warehouses* que estão interligadas a ferramentas para a análise estatística. Uma *data warehouse* é uma base de dados que armazena dados agregados em células multidimensionais, por exemplo, o número de clientes que compraram um tipo de produto num dado período de tempo e numa dada região geográfica.

- *Sistemas de apoio à decisão*. Estes sistemas avaliam decisões através da interacção com pessoas. Existem dois tipos de sistemas de apoio à decisão. O primeiro tipo é baseado em modelos matemáticos. Exemplos incluem sistemas de orçamentação e investimento e sistemas de planeamento de produção. O segundo tipo é baseado em sistemas de raciocínio lógico. São também conhecidos como sistemas periciais. Um exemplo é um sistema para estabelecer a causa de um defeito numa máquina. Estes sistemas são usados em todos os níveis de gestão (tempo real, operacional, táctico e estratégico).
- *Sistemas de controlo*. Também conhecidos por sistemas programados para a tomada de decisão, calculam e implementam decisões de forma completamente automática, com base no estado gravado de um processo. Exemplos incluem encomendas automáticas, controlo climático e sistemas de facturação.

Um sistema de informação é frequentemente uma combinação dos quatro tipos de sistema descritos anteriormente. Do ponto de vista da eficiência, o sistema de controlo parece ser o ideal porque não requer intervenção humana. Na prática, o número de aplicações nas quais podem ser usados tais sistemas é muito limitado, e somente as situações de decisão bem definidas podem ser tratadas desta maneira. No entanto, resolvem alguns problemas da gestão operacional. Os sistemas de apoio à decisão que resolvem problemas de gestão por interacção com pessoas oferecem um maior potencial porque combinam a perspicácia humana com o poder de cálculo do computador. Presentemente, ainda não temos nenhuma ideia de como um sistema de informação deveria tomar uma decisão estratégica sobre muitos problemas. Na prática, a grande maioria dos sistemas de informação são sistemas de escritório e de processamento de transacções.

Examinaremos agora o modo através do qual desenvolvemos sistemas de informação. Isto será feito por meio de um resumo histórico. Os limites dos intervalos de tempo dados não devem ser considerados rígidos, mas isso não é o ponto mais importante. O resumo seguinte destaca a influência dos sistemas de gestão de *workflow*. O que a história mostra é que as tarefas mais genéricas foram gradualmente sendo removidas dos programas e colocadas em sistemas de gestão separados. A figura 1.11 ilustra esta evolução.

1965 – 1975: aplicações separadas. Durante este período, os sistemas de informação consistiam em aplicações separadas, cada uma com as suas próprias bases de dados e definições. As aplicações funcionavam directamente sobre o sistema operativo, e, ou não tinham interface com o utilizador, ou então tinham uma totalmente própria. Os dados eram armazenados entre duas execuções da aplicação, originalmente em cartões perfurados e em fitas de papel, e mais tarde em fitas magnéticas e na memória de discos. Não existia nenhuma troca de

dados entre aplicações diferentes. Assim, era possível que um membro de uma equipa de funcionários tivesse nomes diferentes no programa da folha de pagamentos e no programa de gestão de pessoal. Era impossível obter um valor acrescentado combinando diferentes fontes de dados.

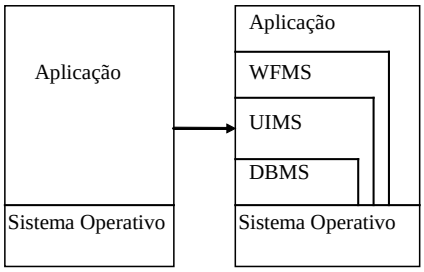


Figura 1.11. Decomposição da funcionalidade genérica

1975 – 1985: gestão de base de dados – "retira a gestão dos dados das aplicações." Este período é caracterizado pela ascensão dos *sistemas de gestão de base de dados* (SGBD). Originalmente estes eram bases de dados hierárquicas e de rede, mais tarde do tipo relacional. Uma base de dados é uma colecção integrada de ficheiros de dados permanentemente disponível que pode ser usada por várias aplicações. O uso de bases de dados tem vantagens pelo facto dos dados geridos por aplicações diferentes poderem ser combinados. As estruturas de dados só precisam de ser definidas uma vez e a organização dos dados pode ser entregue a um sistema de gestão de base de dados. Assim, o mesmo item de dados apenas precisa de ser armazenado uma só vez. Um SGBD é um componente genérico de *software* que pode ser usado para definir e usar bases de dados: para adicionar, ver, rever e apagar dados. O uso de sistemas de gestão de base de dados também mudou radicalmente o processo de desenvolvimento de sistemas: uma vez definida a base de dados, as diferentes equipas de desenvolvimento podem trabalhar no desenho de aplicações ao mesmo tempo. Para fazer isto, foram desenvolvidos métodos para estabelecer estruturas de dados antes das aplicações serem definidas. Esta é a aproximação orientada aos dados no desenvolvimento de um sistema. Este período pode ser assim caracterizado como aquele durante o qual a organização de dados começava a ser extraída das aplicações.

1985 – 1995: gestão da interface com o utilizador – "retira a interface com o utilizador das aplicações." Foi durante este período que o próximo problema no desenvolvimento de sistemas apareceu. Porque estávamos a desenvolver sistemas de *software* cada vez mais interactivo, uma grande parte do tempo estava a ser dispendido a desenvolver interfaces com o utilizador. Originalmente, estas foram desenhadas pelas equipas de desenvolvimento ecrã a ecrã, campo a campo. Isto, não só tomava muito tempo como também cada *designer* tinha o seu próprio estilo, o que significou que cada sistema acabava por operar de uma forma diferente. Existem agora *sistemas de gestão de interfaces com o utilizador* (SGIU) que resolvem estes dois problemas: uma

interface com o utilizador pode ser definida rapidamente e o *designer* é "convidado" a fazê-lo de uma forma *standard*. Recentemente, ocorreu uma transição das interfaces com o utilizador baseadas em caracteres para interfaces gráficas e consequentemente a utilização dos SGIU aumentou. Hoje em dia as funções dos sistemas de gestão de interfaces com o utilizador estão integradas com outras ferramentas, como com os sistemas de gestão de base de dados, ambientes de programação e *web browsers*. Durante este período as interfaces com o utilizador foram extraídas das aplicações.

1995 – 2005: gestão de workflow – "retira os processos de negócio das aplicações." Agora que a gestão dos dados e das interfaces com o utilizador desapareceram da maior parte das aplicações, parece que muito do *software* está dedicado aos processos de negócio (procedimentos) e à manipulação de casos. Consequentemente, tornou-se atractivo isolar este componente, e encontrar uma solução separada para ele. Não só isto pode acelerar o desenvolvimento de sistemas de informação, mas também oferece uma vantagem adicional por tornar os processos de negócio mais fáceis de manter e gerir. Hoje em dia, ocorre frequentemente, que os gestores desejam modificar um procedimento administrativo, mas isso teria consequências significativas para o *software*. Por isso, a mudança não é finalizada. Os *sistemas de workflow* resolvem tais problemas. Um sistema de *workflow* gere os *workflows* e organiza o encaminhamento de dados de um caso entre os recursos humanos e aplicações. Tal como as bases de dados são desenvolvidas e usadas com o auxílio de um sistema de gestão de base de dados, os *sistemas de gestão de workflow* (WFMS) podem ser usados para definir e usar sistemas de *workflow*. Este período pode ser caracterizado como aquele durante o qual os processos foram extraídos das aplicações.

Para colocar a gestão de *workflows* numa perspectiva histórica, devemos mencionar alguns dos primeiros trabalhos nessa área. A ideia de ter ferramentas genéricas, ou pelo menos métodos genéricos, para apoiar os processos de negócio surgiu nos anos 70 com pioneiros tais como Skip Ellis e Michael Zisman. Zisman completou a sua tese de doutoramento intitulada "Representation, Specification and Automation of Office Procedures" em 1977 na Universidade da Pensilvânia. Nos anos 70, Ellis e outros trabalharam para a Xerox PARC em "Office Automation Systems." Ellis já usava modelos de *workflow* baseados em redes Petri (também chamadas de redes de controlo de informação) nos anos 70. Poderíamos querer saber porque levou tanto tempo antes que os sistemas de gestão de *workflow* se estabelecessem como um componente *standard* para sistemas de informação de uma empresa. Existem diversas razões para isso. Em primeiro lugar, a gestão de *workflows* requer que os utilizadores estejam ligados a uma rede de computador. Somente nos anos 90 os trabalhadores passaram a estar ligados em rede. Em segundo lugar, muitos sistemas de informação evoluíram a partir de sistemas que não suportavam processos de negócio e da organização de sistemas que o suportavam; consequentemente, o *workflow* nunca foi considerado como uma parte realmente nova de funcionalidade. Finalmente, o carácter rígido e inflexível dos primeiros produtos (e alguns dos contemporâneos) afastou muitos dos potenciais utilizadores.

Um sistema de gestão de *workflow* pode ser comparado com um sistema operativo: controla os *workflows* entre os vários recursos – pessoas ou aplicações. É confinado à *logística* da manipulação dos casos. Ou seja, uma mudança ao conteúdo dos dados de um caso é implementada somente por pessoas ou por aplicações. Um sistema de gestão de *workflow* tem um número de funções que podem ser usadas para definir e delinear graficamente os *workflows*, tornando tanto o progresso de um caso através de um *workflow* como a estrutura do próprio fluxo, fácil de entender. Consequentemente, não é extraordinário que os sistemas de gestão de *workflow* se tenham tornado na ferramenta ideal para alcançar o BPR (*Business Process Reengineering*).

Na evolução anteriormente descrita, podemos ver que *remover* funções das aplicações é uma forma de melhorar a eficiência. Pela separação de determinadas funções podemos desenvolver soluções (sistemas de gestão) genéricas para esses sistemas. Desta forma os sistemas de informação podem ser feitos *baseados em componentes*, primeiro *configurando* os componentes e depois *integrando-os* (um processo também conhecido por *montagem*). A configuração é a definição de parâmetros que podem tomar várias formas. Exemplos da configuração de componentes incluem: a especificação de um esquema relacional num sistema de gestão de base de dados e a definição de um processo num sistema de gestão de *workflow*.

Para a integração de componentes usamos soluções de *middleware*. Algumas formas de *middleware* são apenas um conjunto de *standards* e de características da linguagem que criam uma estrutura de comunicação em tempo de compilação. Uma outra solução consiste em usar um componente que cuida das necessidades de comunicação de outros componentes.

Juntamente com estes desenvolvimentos também podemos observar a compra por parte das organizações de pacotes de *software standard* para processos específicos que combinam um grande número das funções definidas anteriormente. Para um processo específico, tal *software genérico* tem de ser configurado; isto é, os parâmetros devem ser ajustados. A vantagem de um pacote de *software standard* é que não existem custos de desenvolvimento, mas uma desvantagem é que o sistema pode não possuir todos os requisitos dos seus utilizadores. Esta desvantagem poderia ser vista como um benefício, porque obriga a organização a trabalhar de uma forma já experimentada que está incorporada no pacote de *software*. De facto, tal pacote de *software* contém um *modelo de organização* genérico que pode ser adaptado para situações de negócio específicas.

EXERCÍCIOS

Exercício 1.1

Um *workflow* é definido como uma rede de tarefas com regras que determinam a ordem (parcial) na qual as tarefas devem ser executadas.

(a) Quais são estes princípios essenciais de ordenação?
(b) Mostre que a iteração pode ser feita pelos outros princípios de ordenação.

Exercício 1.2

Neste capítulo vimos (figura 1.2) algumas notações para descrever uma rede de tarefas. (Esta não é a notação que iremos utilizar nos próximos capítulos.) Uma tarefa é representada por um rectângulo e tem um ou mais antecedentes e um ou mais sucessores directos. As regras são: todos os antecedentes têm de estar prontos antes que a tarefa seja executada e exactamente um sucessor será executado. Adicionalmente, existem dois tipos de ligações: os círculos abertos e os círculos fechados com regras para transmitir sinais. Altere essas regras da seguinte forma: as tarefas têm exactamente um arco de chegada e um de saída. As ligações podem ter um ou mais arcos de entrada e de saída. Os círculos abertos transmitem o sinal de um único arco de chegada para exactamente um arco de saída. Os círculos fechados requerem um sinal de todos os arcos de chegada e transmitem-no para todos os arcos de saída. Modele o tratamento de reclamações de acordo com o exemplo da figura 1.2 com estas novas regras. (É permitido ligar círculos entre si.)

Exercício 1.3

O conceito de "tarefa" tem dois significados, dependendo do ponto de vista. Descreva e explique esses dois significados.

Exercício 1.4

Menciona os três princípios para atribuir funcionários aos departamentos de uma organização hierárquica e menciona os prós e os contras para cada escolha.

Capítulo 2.

Modelar Workflows

2.1 Conceitos de Workflow

O sucesso de um sistema de *workflow* depende inteiramente da qualidade dos *workflows* desenhados. Como tal, este livro dedica uma atenção especial à modelação e análise de *workflows*. Neste capítulo, a nossa atenção incidirá inicialmente sobre o conceito de processo. A ferramenta que iremos usar tem por base as redes de Petri. Com a sua ajuda podemos representar um processo de forma clara e concisa. Podemos também usá-las para analisar processos. Falaremos em mais profundidade deste tipo de redes no capítulo 4. Antes disso, devemos em primeiro lugar examinar com mais detalhe alguns dos conceitos introduzidos no capítulo 1.

2.1.1 O caso

O primeiro objectivo de um sistema de *workflow* é lidar com *casos*. Exemplos de casos incluem uma reclamação de um prémio de seguro, um pedido de empréstimo, uma devolução de imposto, uma encomenda ou o tratamento de um paciente num hospital. Casos semelhantes dizem-se pertencer ao mesmo *tipo de caso*. Em princípio, tais casos são tratados da mesma forma.

Cada caso possui uma *identidade* única. Isto torna possível referenciar o caso em questão. Um caso tem uma duração limitada. Considere, por exemplo, a reclamação de um prémio de seguro. Este caso começa no momento em que a reclamação é submetida, desaparecendo do sistema de *workflow* no ponto em que o processamento desta foi concluído. Entre o início e o fim, um caso tem sempre um *estado*. Este estado é composto por três elementos: (1) os valores dos *atributos do caso* relevantes; (2) as *condições* que tenham sido satisfeitas; e (3) o *conteúdo do caso*.

Podemos associar um conjunto de variáveis a cada caso. Os atributos do caso são usados para a sua gestão. Graças a eles, é por exemplo, possível indicar que uma dada tarefa pode – sob certas condições – ser omitida. No tratamento de uma reclamação de seguro, podemos usar o atributo "valor estimado do prémio". Com base no valor desta variável, o sistema de *workflow* pode decidir se activa ou não a tarefa "enviar perito". Note que o valor de um atributo de caso pode mudar a medida que o caso progride.

Não podemos usar um atributo de caso para verificar o progresso de um caso. Para fazê-lo usamos condições. Estas são usadas para determinar quais as tarefas que foram executadas e as que estão por executar. Exemplos de condições são "pedido aceite", "aplicação recusada" e "pedido sob avaliação". Podemos também ver a condição como um requisito que tem de ser cumprido antes que uma tarefa possa ser executada. Só depois de todas as condições para uma determinada tarefa, para um determinado caso, terem sido satisfeitas é que essa tarefa se pode realizar. Para qualquer caso, em qualquer momento, é possível saber que condições foram satisfeitas e as que não o foram. Podemos também usar a palavra *fase* em vez de condição. Esta no entanto pode ser confusa quando várias condições foram satisfeitas: o caso estaria em mais do que uma fase em simultâneo.

Geralmente, o sistema de *workflow* não contém detalhes sobre o conteúdo do caso, apenas contém os detalhes dos seus atributos e condições. Este conteúdo é composto por documentos, ficheiros, arquivos, e/ou bases de dados que não são geridos pelo sistema de gestão de *workflow*.

2.1.2 A tarefa

O termo *tarefa* já foi mencionado extensivamente. É um dos mais importantes conceitos deste livro. Ao identificar as tarefas é possível estruturar os *workflows*. Uma tarefa constitui uma unidade lógica de trabalho, é indivisível e, como tal, é sempre executada por inteiro. Se algo corre mal durante a execução de uma tarefa então temos de retornar ao início da mesma. A isto chamamos de *rollback*. No entanto, esta natureza indivisível da tarefa depende do contexto dentro do qual ela está definida. Uma tarefa que é contratada por um cliente a um fornecedor é encarada como "atómica" (indivisível) do ponto de vista do primeiro (o cliente). Pode não ser esta a visão do fornecedor: ele pode, se quiser, dividir a tarefa em tarefas mais pequenas.

Dactilografar uma carta, avaliar um relatório, registar uma queixa, selar um documento e verificar dados pessoais são exemplos de tarefas. Podemos diferenciá-las em tarefas *manuais*, *automáticas* e *semi-automáticas*. Uma tarefa manual é inteiramente executada por uma ou mais pessoas sem recurso a uma aplicação: por exemplo, fazer um teste médico. Em oposição, uma tarefa automática é executada sem qualquer intervenção humana. Isto significa que uma aplicação – um programa de computador – pode executar a tarefa baseando-se inteiramente em dados armazenados. Pessoas e aplicações participam na realização de tarefas semi-automáticas. Por exemplo, o preenchimento de um relatório por um perito de seguro, assistido por um programa especialmente criado para a tarefa.

Uma tarefa diz respeito a uma parte genérica do trabalho, e não ao desempenho de uma dada actividade de um caso particular. Para evitar a confusão entre a tarefa propriamente dita e o desempenho dessa tarefa como parte de um caso, usamos os termos *item de trabalho* e *actividade*. Um item de trabalho é a combinação entre um caso e uma tarefa prestes a ser executada. Um item de trabalho é criado logo que o estado do caso o permita. Podemos assim ver um item de trabalho como um pedaço concreto de trabalho que pode ser executado. O termo actividade refere-se ao desempenho de um item de trabalho. Assim que o trabalho começa sobre um item de trabalho, este

transforma numa actividade. Notemos que, ao contrário de uma tarefa, tanto o item de trabalho como a actividade estão ligados a um caso específico. O diagrama da figura 2.1 ilustra as relações entre os novos termos introduzidos.

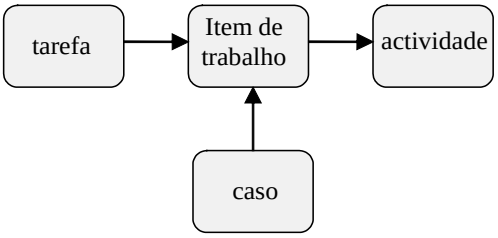


Figura 2.1. Relação entre os termos tarefa, caso, item de trabalho e actividade

2.1.3 O processo

A forma pela qual uma determinada categoria de casos deve ser executada é descrita pelo *processo* relevante e associado. Este indica quais as tarefas que necessitam de ser executadas. Também indica a ordem pela qual devem ser executadas. Podemos também olhar para um processo como um *procedimento* para um *tipo de caso* particular. Em geral, muitos casos diferentes são tratados usando um único processo. Isso possibilita um tratamento específico com base nos atributos de um caso. Por exemplo, podemos ter uma tarefa, de um processo, que seja executada apenas para alguns dos casos. A ordem pela qual as tarefas são executadas pode também variar, dependendo das propriedades do caso. Usam-se condições para decidir qual a ordem a seguir. Essencialmente, um processo é construído a partir de tarefas e de condições.

É possível reutilizar processos previamente definidos como partes de outro processo. Por isso, em adição às tarefas e condições, um processo pode ser constituído por (zero ou mais) sub-processos. Cada sub-processo por sua vez é constituído por tarefas, condições e possivelmente mais processos. Ao identificar explicitamente e descrever separadamente sub-processos, podemos reutilizar aqueles que forem mais úteis e frequentes. Desta forma, os processos complexos podem ser estruturados numa hierarquia. No nível mais alto da descrição do processo podemos visualizar um número limitado de sub-processos. Olhando com mais atenção a um, ou mais, destes sub-processos podemos fazer um *zoom in* nas partes do processo nas quais temos um maior interesse.

O ciclo de vida de um caso é definido por um processo. Como cada caso tem uma duração finita, com um início e fim bem definidos, é importante que o processo esteja em conformidade com o mesmo. Assim, cada processo também tem um início e um fim, os quais marcam respectivamente o aparecimento e a conclusão de um caso.

2.1.4 Encaminhamento

O ciclo de vida de um caso é especificado por um processo. A este respeito referimo-nos ao *encaminhamento* de um caso. Encaminhar um caso ao longo de ramos particulares determina que tarefas precisam de ser executadas (e em que ordem). No encaminhamento de casos fazemos usos de quatro construções básicas:

- A forma mais simples de encaminhamento é a execução *sequencial* de tarefas. Por outras palavras, nesta construção as tarefas são executadas uma a seguir à outra. Existe usualmente uma óbvia dependência entre elas. Por exemplo, o resultado (saída) de uma tarefa é a entrada da próxima tarefa.
- Se duas tarefas podem ser executadas em simultâneo, ou em qualquer ordem, então falamos de encaminhamento *paralelo*. Neste caso, existem *n* tarefas que precisam de ser executadas sem que o resultado de uma afecte o resultado da outra. As *n* tarefas são iniciadas usando um *AND-split* e mais tarde re-sincronizadas usando um *AND-join*.
- Falamos de encaminhamento *selectivo* quando é possível escolher entre duas ou mais tarefas. Esta escolha pode ser dependente de propriedades específicas do caso que são registadas nos seus atributos. Tal escolha entre alternativas é conhecida por *OR-split*. Os ramos alternativos são reunidos usando um *OR-join*. Também podemos usar os termos encaminhamento alternativo ou condicional.
- Numa situação ideal, uma tarefa é executada não mais de uma vez por caso. No entanto, existem situações em que é necessário executar uma tarefa várias vezes. Consideremos, por exemplo, uma tarefa que tem de ser repetida até que o resultado da subsequente tarefa de verificação seja satisfatório. Chamamos a esta forma de encaminhamento *iteração*.

Voltaremos a estas quatro formas de encaminhamento em detalhe mais à frente.

2.1.5 Execução

Um item de trabalho só pode ser executado quando o estado do caso em questão o permita. Mas na prática, o desempenho desse processamento requer frequentemente mais do que isso. Se tem de ser processado por uma pessoa, esta tem de retirar o item de trabalho da sua "caixa de entrada" antes de iniciar a actividade. Por outras palavras, o item de trabalho é trabalhado a partir do momento em que o funcionário toma a iniciativa. Este caso é referido como *triggering* (disparo): o item de trabalho é iniciado por um recurso (no nosso exemplo, o empregado). No entanto, outras formas de disparo são possíveis: um evento externo (por exemplo a chegada de uma mensagem EDI – *Electronic Data Interchange*) ou a passagem de uma hora específica (por exemplo, a geração de uma lista de pedidos às seis horas em ponto). Distinguimos assim três tipos de disparos: (1) por iniciativa de um recurso, (2) um evento externo e (3) um sinal temporal. Os itens de trabalho cujo processamento tem de ser

efectuado de imediato – sem a intervenção de estímulos externos – não precisam de um disparo.

Os conceitos acima resumidos são o tema central deste capítulo. Teremos uma ênfase específica sobre a modelação de processos que estão subjacentes aos *workflows*. No próximo capítulo falaremos da reserva de itens de trabalho, o arranjo da estrutura organizacional e as aptidões específicas dos funcionários. No capítulo 4 veremos como podemos analisar *workflows* depois de modelados.

2.2 Redes de Petri

Ao contrário de muitas outras publicações sobre a gestão de *workflows*, este livro faz uma abordagem formal baseada num formalismo já estabelecido para a modelação e análise de processos – as redes de Petri. O uso de conceitos formais tem um grande número de vantagens. Em primeiro lugar, obriga a uma definição precisa. Prevenindo ambiguidades, incertezas e contradições, como acontece em muitas técnicas informais de desenho. Por outro lado, o formalismo pode ser usado para discutir os processos. Torna-se assim possível, por exemplo, estabelecer certos padrões. Isto está intimamente ligado ao facto de que um formalismo permite o uso de uma série de técnicas analíticas (aquelas para analisar o desempenho, por exemplo, bem como aquelas para verificar propriedades lógicas). Como veremos mais adiante, torna-se possível verificar se um caso é completado com sucesso após um certo período de tempo. Existem por isso várias boas razões para optar por um método formal. Antes de expormos os conceitos anteriormente enunciados neste capítulo é importante saber um pouco mais sobre esta técnica, as redes de Petri.

As redes de Petri foram criadas em 1962 por Carl Adam Petri como ferramenta para modelar e analisar processos. Um dos pontos fortes desta ferramenta é o facto de possibilitar a descrição *gráfica* de processos. Depois veremos que podemos usá-la para representar processos de *workflow* de uma forma simples e acessível. Embora as redes de Petri sejam gráficas, elas têm uma sólida base matemática. Ao contrário de tantas outras técnicas esquemáticas, as redes de Petri são totalmente formalizadas e, graças a isto, é possível fazer afirmações seguras acerca das propriedades dos processos modelados. Existem também várias técnicas de análise e ferramentas disponíveis que podem ser aplicadas para analisar uma determinada rede de Petri.

Ao longo dos anos, o modelo proposto por Carl Adam Petri tem sido extendido de várias formas. Graças a estas é possível hoje modelar processos complexos de uma maneira acessível. Inicialmente, no entanto, iremos limitarmos-nos às redes de Petri clássicas tal como foram criadas pelo próprio Petri.

2.2.1 Redes de Petri clássicas

Uma *rede de Petri* é composta por *lugares* e *transições*. Assinalamos um lugar com um círculo. Uma transição é ilustrada com um rectângulo. A figura 2.2 mostra uma rede de Petri simples, constituída por três lugares (*reclamação*, *em_avaliação* e *pronto*) e por três transições (*registar*, *pagar* e *enviar_carta*). Esta rede modela o processo de uma reclamação de prémio de seguro. A

chegada ao lugar *reclamação* é primeiro registada, após a qual é feito o pagamento ou é enviada uma carta explicando as razões da rejeição.

Os lugares e as transições numa rede de Petri estão ligados por meio de *arcos direccionados*. Na figura 2.2, por exemplo, o lugar *reclamação* e a transição *registar* estão ligadas por uma seta que aponta do primeiro para o segundo elemento. Existem dois tipos de arcos: os que saem de lugar para uma transição e os que saem de uma transição para um lugar. Arcos de lugar para lugar, ou arcos de transição para transição, não são possíveis.

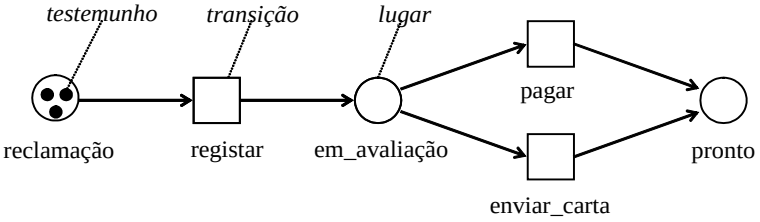


Figura 2.2. Uma rede de Petri clássica

Com base nos arcos podemos determinar os *lugares de entrada* para uma transição. Um lugar *l* é um lugar de entrada para uma transição *t*, se – e só se – existir um arco direccionado de *l* para *t*. De igual modo, podemos determinar os *lugares de saída* para uma transição. Um lugar *l* é um lugar de saída de uma transição *t* se – e só se – existe um arco direccionado de *t* para *l*. Como acontece na figura 2.2, cada transição tem precisamente um lugar de entrada e um lugar de saída.

Os lugares podem conter *testemunhos*. Estes são indicados usando pontos pretos. Na figura 2.2, o lugar *reclamação* contém três testemunhos. A estrutura de uma rede de Petri é fixa. No entanto, a distribuição dos testemunhos por lugares pode mudar. A transição *registar* pode assim pegar em testemunhos do lugar de entrada (*reclamação*) e colocá-los no lugar *em_avaliação*. A esta acção, denominamos de *disparo* da transição *registar*. Porque o disparo de uma transição está sujeito a regras estritas, iremos primeiro introduzir um conjunto de termos relevantes.

O estado de uma rede de Petri é indicado pela distribuição dos testemunhos pelos seus lugares. Podemos descrever o estado ilustrado na figura 2.2 usando o vector (3,0,0). Por outras palavras, existem três testemunhos no lugar reclamação, zero testemunhos em *em_avaliação*, e zero em *pronto*.

Uma transição só pode disparar se estiver *pronta*. Isto ocorre quando existe pelo menos um testemunho em *cada* um dos seus lugares de entrada. As transições estão "carregadas", i.e. prontas a disparar. Na figura 2.2, a transição *registo* está pronta. As outras duas não estão.

Uma transição pode disparar a partir do momento em que esteja *pronta*. Quando dispara, um testemunho é removido de um dos lugares de entrada e um testemunho é adicionado a cada um dos lugares de saída. Por outras palavras, no momento do disparo, uma transição *consome* testemunhos dos lugares de entrada e *produz* testemunhos para os lugares de saída. A figura 2.3 mostra o efeito do disparo da transição *registo*. Como resultado, um testemunho

é transferido do lugar *reclamação* para o lugar *em_avaliação*. Podemos também descrever esta nova situação usando o vector (2,1,0).

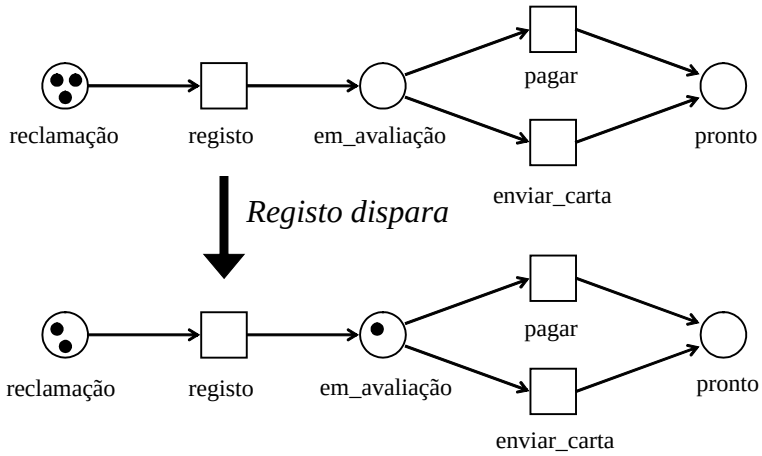


Figura 2.3. O estado antes e depois de disparar a transição "registar"

Depois da transição *registo* disparar, surge uma situação em que três transições estão prontas. A transição *registo* pode disparar mais uma vez porque existe pelo menos um testemunho no lugar *reclamação*, e as transições *pagar* e *enviar_carta* podem disparar porque existe pelo menos um testemunho no lugar *em_avaliação*. Nesta situação, não é possível determinar qual das duas transições irá disparar. Se assumirmos – por conveniência – que é a transição *pagar* que irá disparar, então será atingido o estado ilustrado na figura 2.4.

Notemos que a transição *enviar_carta*, que estava pronta antes do disparo, já não se encontra pronta. A transição *registo* ainda está pronta e disparará. Eventualmente, após um total de seis disparos, a rede de Petri chegará ao estado (0,0,3). Isto é, um estado com três testemunhos no lugar *pronto*. Neste estado, não é possível realizar mais nenhum disparo.

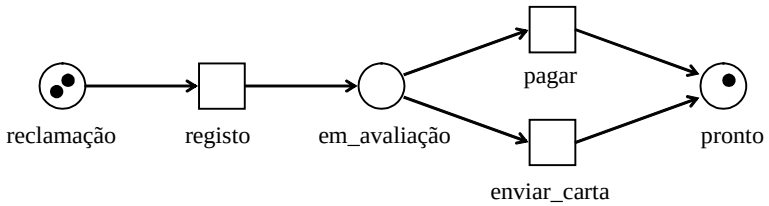


Figura 2.4. Estado após a transição "pagar" disparar

As transições são os componentes *activos* numa rede de Petri. Ao disparar uma transição, o processo modelado muda de um estado para outro. Uma transição representa assim um evento, uma operação, uma transformação, ou

um transporte. Os lugares numa rede de Petri são *passivos*, no sentido em que eles não podem alterar o estado da rede. Um lugar representa um meio, um acumulador, uma localização geográfica, um sub-estado, uma fase, ou uma condição. Os testemunhos geralmente representam objectos. Estes podem ser físicos, mas também representativos de informação. Na rede acima considerada, cada testemunho representa uma reclamação de seguro.

Na rede de Petri ilustrada na figura 2.2 é possível que vários casos progridam em simultâneo. Se a transição *registo* dispara duas vezes sucessivamente, então estarão pelo menos dois testemunhos no lugar *em_avaliação*. Se, por alguma razão, queremos limitar o número de casos que podem estar em avaliação ao mesmo tempo para o máximo de um, podemos modificar a rede Petri como mostra a figura 2.5. O novo lugar *livre* assegura que a transição *registo* está bloqueada logo que a reclamação passa a estar em avaliação.

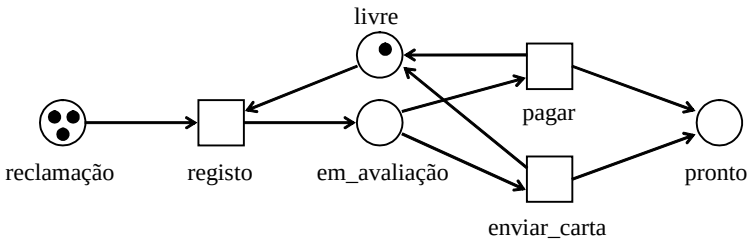


Figura 2.5. A rede de Petri modificada

No estado inicial (figura 2.5), a transição *registo* está pronta pois existe pelo menos um testemunho em cada um dos lugares de entrada. Quando a transição *registo* tiver disparado, passa para um estado no qual *registo* não se encontra pronto, mas as outras duas transições estão. Quando uma destas disparar, passa a existir um testemunho no lugar *livre*. Só neste momento é que *registo* volta a estar pronta. Ao adicionarmos o lugar *livre*, o número máximo de casos que podem estar a ser avaliados em qualquer altura foi efectivamente limitado a um. Se quisermos limitar o número de casos em progresso em qualquer altura para um máximo de *n*, então podemos modelar esta situação colocando *n* testemunhos no lugar *livre*.

Usando redes de Petri, podemos descrever processos que tenham uma natureza repetitiva. A figura 2.6 mostra como podemos modelar a actividade cíclica de um conjunto de três luzes de um semáforo.

As três configurações possíveis das luzes dos semáforos são representadas por três lugares: *vermelho*, *amarelo* e *verde*. As três mudanças possíveis entre as luzes são representadas pelas transições: *vv*, *va* e *av*. Imaginemos que queremos modelar o sistema de luzes no cruzamento de duas vias de sentido único. Neste caso, precisamos de dois semáforos que interajam de tal forma que um deles esteja sempre a vermelho. Obviamente, a rede de Petri mostrada na figura 2.6 precisa de ser duplicada. Cada conjunto de luzes é modelado usando três lugares e três transições. Isto, no entanto não é suficiente porque não exclui situações inseguras. Assim, adicionamos um lugar extra *x*, que assegura que um dos dois semáforos está sempre a vermelho (ver figura 2.7).

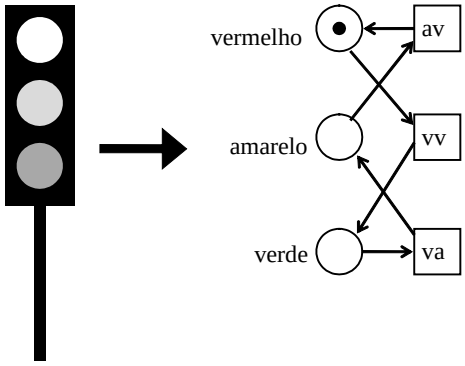


Figura 2.6. Um semáforo representado por uma rede de Petri

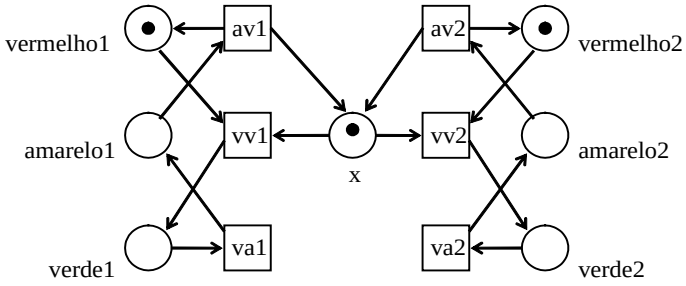


Figura 2.7. Dois semáforos de trânsito

Quando ambas as luzes estão vermelhas, existe um testemunho no lugar *x*. Logo que um dos semáforos muda para verde, o testemunho é removido de *x*, ficando o outro semáforo bloqueado. Só quando estiverem novamente ambos a vermelho é que o outro poderá mudar para verde. No capítulo 4 iremos usar uma técnica analítica para mostrar que os semáforos funcionam de forma segura.

2.2.2 Redes de Petri de alto nível

Porque as redes de Petri são gráficas, elas são fáceis de utilizar. Também têm uma sólida base matemática e existem muitas técnicas analíticas disponíveis. No capítulo 4, veremos que podemos usar estas técnicas para analisar *workflows*. Embora este seja um ponto forte, a rede de Petri clássica tem limitações em muitas situações práticas, pois torna-se demasiado grande e inacessível, ou então não é possível modelar uma actividade específica. Por causa disto, a rede de Petri clássica foi estendida de várias formas. Graças a estas extensões é possível modelar situações complexas de forma estruturada e acessível. Nesta secção

focaremos sobre as três mais importantes extensões: (a) extensão de cor, (b) extensão de tempo e (c) extensão hierárquica. Chamamos às redes de Petri estendidas com cor, tempo e hierarquia, *redes de Petri de alto nível*. Como uma descrição completa das características das redes de Petri de alto nível seria demasiado longa, iremos limitar a nossa abordagem apenas aos aspectos mais importantes no contexto da gestão de *workflows*.

(a) **Extensão por cor**. Os testemunhos são usados para modelar todo um conjunto de coisas. Num modelo podemos representar reclamações de seguro, noutro o estado das luzes de um semáforo. No entanto, numa rede de Petri clássica é impossível distinguir entre dois testemunhos: dois testemunhos no mesmo lugar são, por definição, indistinguíveis. Esta é uma situação indesejável. No caso de duas reclamações de seguro, por exemplo, podemos querer incorporar características separadas das duas reclamações no modelo. Queremos incluir aspectos como a natureza da reclamação, o número da apólice, o nome do titular do seguro e o valor calculado da reclamação. Para permitir a associação das características de um objecto com o testemunho correspondente, a rede de Petri clássica é estendida usando cores. Esta extensão assegura que a cada testemunho é atribuído um *valor* ou *cor*. Um testemunho que representa um determinado carro terá um valor que torna possível identificar a marca, o registo de propriedade, o ano de fabrico, a cor e o dono. Podemos representar um possível valor para tal testemunho da seguinte forma: [*lugar: "BMW"; registo: "J 144 NFX"; ano: "1995"; cor: "vermelho"; dono: "Johnson"*].

Como cada testemunho tem um valor podemos distinguir testemunhos diferentes entre si. Ao atribuir um valor aos testemunhos é como se estes recebessem cores distintas.

O disparo de uma transição produz testemunhos baseados nos valores daqueles consumidos durante o disparo. Por isso, o valor de um testemunho produzido pode depender dos testemunhos consumidos. Ao contrário da rede de Petri clássica, o *número* de testemunhos produzidos é determinado pelos valores dos consumidos.

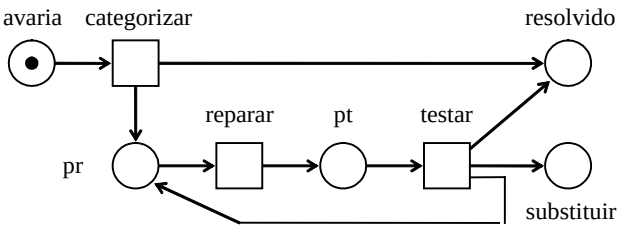


Figura 2.8. O processo de resolução de avarias

Para ilustrar esta situação, usaremos um processo para resolver avarias técnicas num departamento de produtos. Sempre que ocorre uma avaria – por exemplo, uma máquina encravada – esta é categorizada pelo mecânico do departamento. A avaria pode muitas vezes ser resolvida no momento da sua categorização. Se não for esse o caso, então é levado a cabo uma reparação. Depois da reparação é feito um teste com três possíveis resultados: (1) a avaria foi resolvida; (2) é necessária uma reparação adicional; ou (3) a peça estragada

tem de ser substituída. Este processo é modelado na figura 2.8 usando uma rede de Petri.

Um testemunho no lugar *avaria* significa que uma avaria ocorreu e que precisa de ser tratada. Para cada testemunho em *avaria* a transição *categorizar* disparará precisamente uma vez. A cada disparo, um testemunho será produzido e colocado no lugar *resolvido* ou no lugar *pr* (precisa de reparação). Contrariamente à rede de Petri clássica, é agora possível a um lugar de saída não receber um testemunho. Durante a execução da transição *categorizar*, a escolha é agora feita com base na informação disponível. Como resultado desta escolha, a avaria é interpretada como tendo sido resolvida ou então é feita uma reparação. O testemunho no lugar *avaria* tem um valor no qual as propriedades relevantes da avaria são registadas (por exemplo: a natureza da avaria, a identidade do componente que não funciona, o código de localização e o histórico de avarias). Se uma reparação for necessária, então a transição *reparar* dispara, trazendo um testemunho ao lugar *pt* (precisa de teste), seguida pela transição *testar*. A transição *testar* produz um testemunho que aparece num de três lugares de saída. A informação relevante da avaria é sempre mantida no valor do testemunho em questão.

Numa rede de Petri estendida por cor *podemos* fixar condições para os valores dos testemunhos a serem consumidos. Se é este o caso, então uma transição fica apenas pronta logo que haja um testemunho em cada um dos lugares de entrada e as *pré-condições* tenham sido cumpridas. A pré-condição de uma transição é um requisito lógico associado aos valores dos testemunhos a serem consumidos. Na rede de Petri ilustrada na figura 2.8 podíamos, por exemplo, acrescentar a seguinte pré-condição à transição *categorizar*: "O valor do testemunho a ser consumido do lugar *avaria* tem de conter um código de localização válido." A consequência desta pré-condição é que as avarias sem localização válida não chegam a ser categorizadas; estas permanecem no lugar *avaria* e nunca são consumidas pela transição *categorizar*.

Podemos também usar uma pré-condição para "sincronizar" testemunhos. Desta forma podemos indicar que uma transição apenas dispara se uma combinação específica de testemunhos puder ser consumida. Usamos para tal a transição *montar*, ilustrada na figura 2.9.

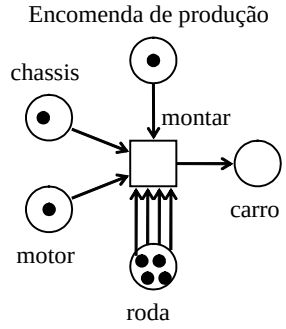


Figura 2.9. A transição "montar"

Com base num pedido de produção, a transição *montar* precisa de um chassis, de um motor, e de quatro rodas para produzir um carro. (Este é o primeiro exemplo que vemos, no qual mais do que um arco sai de um ponto de entrada para uma transição. Neste caso, têm de existir pelo menos quatro testemunhos no lugar *roda* antes que a transição *montar* possa estar pronta. O número de arcos de entrada mostra quantos testemunhos têm de existir no ponto de entrada. Quando uma transição dispara, o número de testemunhos consumidos é igual ao número de arcos de entrada). Quando a transição *montar* dispara, os testemunhos não são levados aleatoriamente dos lugares de entrada. Por exemplo, as quatro rodas têm de ser do mesmo tipo, o motor tem que encaixar no chassis, o diâmetro das quatro rodas tem de ser adequado ao chassis e à potência do motor, e assim por diante. Os testemunhos só são retirados dos lugares de entrada em certas combinações. Isto é determinado por meio de uma pré-condição.

O resultado da extensão de cor é que, ao contrário das redes de Petri clássicas, a representação gráfica já não contém toda a informação. Para cada transição, os seguintes factores têm de ser especificados:

- Se existe uma pré-condição, então esta tem de ser definida com precisão.
- O número de testemunhos produzidos por cada ponto de saída durante cada disparo. Este número pode depender dos valores dos testemunhos consumidos.
- Os valores dos testemunhos produzidos podem também depender dos valores dos testemunhos consumidos.

Dependendo do objectivo para o qual foi feita a rede de Petri, as transições são especificadas com texto, algumas linhas em pseudo-código, uma especificação formal ou com uma sub-rotina numa linguagem de programação.

(b) **Extensão por tempo**. Dado um processo modelado com uma rede de Petri, queremos muitas vezes poder fazer afirmações sobre o seu desempenho previsto. Se produzimos um modelo de semáforos num cruzamento, então provavelmente também estamos interessados no número de veículos que o cruzamento pode escoar por hora. Se modelarmos um processo de produção de uma fábrica de automóveis então também queremos saber o tempo que demora a completar um carro e a capacidade das instalações. Para poder responder a estas questões é necessário incluir informação pertinente sobre a duração de um processo no modelo. No entanto, a rede de Petri clássica não permite a modelação do "tempo". Mesmo com a extensão de cor é ainda assim difícil modelar a duração do processo. Por isso, a rede de Petri clássica foi também estendida com *tempo*.

Usando esta extensão, os testemunhos recebem um *timestamp* (referência temporal) e um valor. Isto indica o tempo a partir do qual o testemunho está disponível. Um testemunho com um *timestamp* de 14 está assim disponível para consumo por uma transição apenas a partir do momento 14. Uma transição está pronta apenas no momento em que cada um dos testemunhos a serem consumidos tem um *timestamp* igual ou anterior ao tempo actual. Por outras palavras, o tempo que torna uma transição *pronta* (*enabling time*) é o primeiro momento a partir do qual os seus lugares de entrada contêm testemunhos suficientes *disponíveis*. Os testemunhos são consumidos usando um esquema

FIFO (*first-in*, *first-out*). O testemunho com o *timestamp* mais antigo será o primeiro a ser consumido. Adicionalmente, é a transição pronta e com o tempo mais antigo que dispara primeiro. Se existir mais de uma transição com o mesmo tempo, a escolha é não determinística. O disparo de uma transição pode afectar o tempo em que outra transição fica pronta.

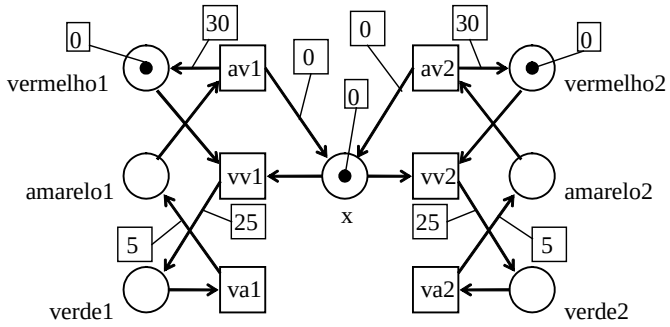


Figura 2.10. Os dois semáforos de trânsito com tempo

Se uma transição dispara e produz testemunhos, então cada um deles recebe um timestamp igual ou superior ao tempo de disparo. Os testemunhos produzidos recebem um *atraso* que é determinado pela transição que disparou. O *timestamp* de um testemunho produzido é igual ao instante do disparo mais esse atraso. A duração do atraso pode depender do valor dos testemunhos consumidos. No entanto, também é possível que o atraso tenha um valor fixo (por exemplo, 0) ou que este seja decidido aleatoriamente. Um auto disparo é instantâneo e não consome tempo.

Para ilustrar a extensão por tempo, podemos usar o exemplo de dois conjuntos de semáforos que não podem estar verdes ou amarelos em simultâneo. No momento zero ambos os semáforos estão a vermelho. Como podemos ver na figura 2.10, o *timestamp* dos testemunhos nos lugares *vermelho1*, *x* , e *vermelho2* é 0.

O tempo em que transição *vv1* está pronta é também 0, o máximo dos *timestamps* dos testemunhos em *vermelho1* e *x*. O tempo em que *vv2* está pronta é também 0. Existe por isso um momento de escolha não determinística entre *vv1* e *vv2*. Suponhamos que *vv1* dispara. A transição *vv1* consome os dois testemunhos dos lugares de entrada e produz um testemunho para o lugar *verde1* com um atraso de 25 unidades de tempo. Na figura 2.10, cada atraso é ilustrado com o auxílio de uma etiqueta associada aos arcos. (Se os atrasos fossem dependentes dos valores dos testemunhos consumidos, isso não seria possível). Depois do disparo de *vv1* existe um testemunho em *verde1* com um *timestamp* de 25 e *va1* é a única transição que está pronta. A transição *va1* dispara no momento 25 e produz um testemunho em *amarelo1* com um *timestamp* igual a 25+5 = 30. No momento 30, a transição *av1* disparará. Durante esse disparo, *av1* produz um *testemunho* para *vermelho1* com um atraso 30 e um testemunho para *x* sem atraso. Como resultado do disparo, a transição *vv1* fica pronta no momento 60 e *vv2* no tempo de 30. Por isso, a transição *vv2* será

a próxima a disparar. Ao acrescentar tempo ao modelo, especificámos não só a duração das várias fases mas também forçámos os semáforos a mudarem para verde alternadamente.

(c) **A extensão hierárquica**. Embora já possamos descrever processos muito complexos usando as extensões de cor e tempo, normalmente a rede de Petri resultante ainda não reflecte adequadamente os processos que necessitam de ser modelados. Porque a modelação de alguns processos resulta numa única e extensiva rede, perdemos informação sobre a sua estrutura. Até agora nunca nos preocupamos com a *estrutura hierárquica* dos processos a serem modelados usando redes de Petri. A extensão hierárquica assegura que é possível adicionar uma estrutura ao modelo de uma rede de Petri.

De modo a estruturar hierarquicamente uma rede de Petri, introduzimos um novo "bloco construtivo": um quadrado com fronteira dupla. Chamamos a este elemento um *processo* que representa uma sub-rede que contém lugares, transições, arcos, e sub-processos. E porque um processo pode ser construído por sub-processos que por sua vez também podem ser constituídos por sub-processos, é possível estruturar um processo complexo hierarquicamente. Para ilustrá-lo, usamos o processo modelado na figura 2.8. Neste processo, a hierarquia está representada na actividade *reparar*. Já não queremos ver a actividade reparar como uma única e indivisível acção, mas como um sub-processo que consiste nos passos seguintes: (1) início, (2) rastrear, (3) mudar, e (4) finalizar. Adicionalmente, nunca existe mais do que uma avaria em reparação num dado momento. Para modelar este detalhe, substituímos a transição *reparar* por um sub-processo constituído por quatro transições e quatro lugares – ver a figura 2.11.

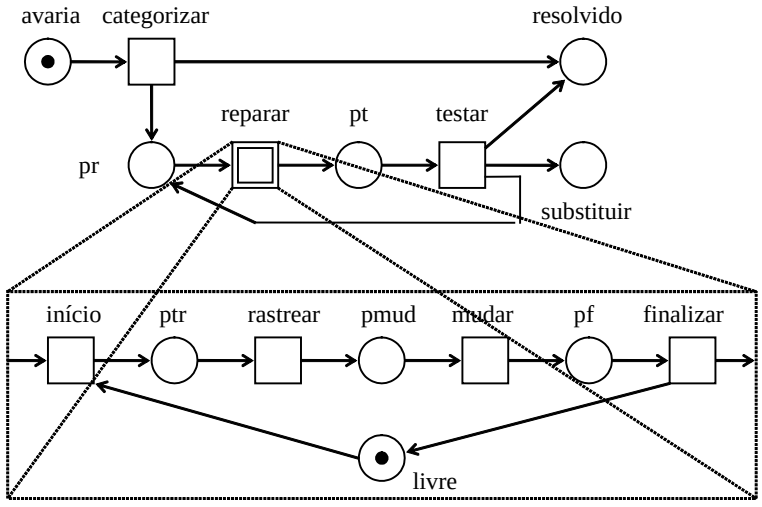


Figura 2.11. O processo "resolver avaria" contém um sub-processo: "reparar"

Na figura 2.11 podemos ver claramente que um processo pode tomar duas formas: (1) como um sub-processo dentro de um processo hierarquicamente superior (o quadrado de fronteira dupla) e (2) como a definição de um processo

(um resumo dos elementos a partir dos quais o processo é construído). Encontramos o significado do processo construído por sub-processos substituindo cada um deles pela sua definição. O processo *resolver_avaria* ilustrado na figura 2.11 é assim uma rede de Petri consistindo de seis transições e nove lugares.

Usando (sub)processos podemos estruturar uma rede de Petri hierarquicamente, usando uma abordagem *top-down* ou *bottom-up*. A abordagem *top-down* começa no nível mais elevado, com cada processo a ser dividido em sub-processos até que, no nível mais inferior, estes consistam apenas por transições e lugares. A decomposição repetida resulta numa descrição hierárquica. A abordagem *bottom-up* funciona na direcção oposta. Começa no nível mais inferior. Primeiro, os componentes mais elementares são descritos em detalhe. Estes elementos (sub-processos) são depois combinados em processos maiores. A composição repetida resulta eventualmente numa descrição do processo global.

Quando modelamos processos complexos é frequentemente necessário usar um método de descrição hierárquico. Só dividindo o processo principal em sub-processos cada vez mais pequenos é que podemos controlar a sua complexidade. A este respeito, referimos a estratégia *dividir-para-conquistar*. No entanto, a identificação de sub-processos tem outra vantagem importante pois possibilita-nos *reutilizar* processos previamente definidos. Se um processo específico ocorre várias vezes, uma só definição usada repetidamente é suficiente. A reutilização de (sub)processos torna frequentemente possível modelar um processo complexo mais rapidamente.

Nesta secção, estudámos os três mais importantes tipos de extensão das redes de Petri: (a) extensão por cor, (b) extensão por tempo, e (c) a extensão hierárquica. Chamamos às redes de Petri que incorporam estas extensões, *redes de Petri de alto nível*. Nos seguintes capítulos, usaremos as redes de alto nível para modelar e analisar processos no contexto da gestão de *workflows*.

2.3 Mapeamento de Conceitos de Workflow para Redes de Petri

Vamos agora passar a demonstrar os conceitos descritos anteriormente – caso, tarefa, condição, processo, disparo e assim sucessivamente – utilizando redes de Petri.

2.3.1 O processo

Através da utilização de um processo num sistema de gestão de *workflow*, podemos indicar de que forma é que uma categoria específica de casos deve ser tratada. O processo define que tarefas precisam de ser executadas. Da mesma forma como contêm informação necessária sobre as tarefas a serem executadas, um processo também contem informação sobre condições que necessitam de ser respeitadas. Desta forma, ele define a ordem pela qual as tarefas precisam de ser executadas. Também é possível usar processos definidos anteriormente dentro de processos de maior porte ou de maior escala. Logo, um processo pode consistir em mais do que um sub-processo, bem como de tarefas ou condições. Por consequência, torna-se óbvio *especificar um processo utilizando uma rede*

de Petri. Esta rede deverá possuir uma "entrada" (um lugar que não possui arcos de entrada) e uma "saída" (um lugar que não possui arcos de saída). Conseguimos mostrar as condições através de lugares e as tarefas usando transições. Isto também é óbvio, visto que as transições são os elementos activos numa rede de Petri, enquanto que os lugares são os seus componentes passivos.

De forma a especificar um processo utilizando uma rede de Petri, iremos examinar um processo criado para tratar reclamações de uma dada empresa. Quando uma reclamação é recebida, primeiro é registada e armazenada. A seguir, o cliente que efectuou a reclamação e o departamento afectado pela reclamação são contactados. O cliente é então contactado para que se possa obter mais informação. O departamento é também informado acerca da reclamação e pode ter de justificar a sua reacção inicial (que conduziu à reclamação do cliente). Estas duas tarefas podem ser executadas em paralelo – isto é, de forma simultânea ou seguindo uma qualquer ordem. Depois os dados são reunidos e uma decisão é tomada. Dependendo da decisão, pode ser feita uma compensação monetária ou pode ser enviada uma carta ao cliente. Finalmente, a queixa é arquivada. A figura 2.12 mostra como podemos ilustrar o processo anteriormente descrito utilizando uma rede de Petri.

Cada uma das tarefas *armazenar*, *contactar_cliente*, *contactar_departamento*, *pagar* e *arquivar* são modeladas utilizando uma transição. A análise de uma reclamação é modelada recorrendo a dois tipos de transições: *positiva* e *negativa*. A transição *positiva* corresponde a uma decisão positiva, a transição *negativa* corresponde a uma decisão negativa (Mais tarde iremos ver como é que esta tarefa pode também ser modelada utilizando simplesmente uma transição.) Os lugares *início* e *fim* correspondem ao início e ao fim do processo que se encontra a ser modelado. Os restantes lugares correspondem a condições que são ou não satisfeitas por cada caso em progresso. As condições têm dois papéis fundamentais: por um lado elas asseguram que as tarefas são encaminhadas na sequência correcta, e por outro lado, garantem que o estado de um caso possa ser determinado. O lugar *c8*, por exemplo, assegura que uma reclamação é arquivada só quando foi completamente resolvida. Também corresponde ao estado existente entre a resolução completa de uma reclamação e o seu posterior arquivo.

Da figura 2.12, é mais ou menos claro que um caso é representado por um ou mais testemunhos. Assim sendo, os casos são ilustrados através da utilização de testemunhos. Na figura, o testemunho no lugar *início* mostra a presença de um caso. Assim que a transição *armazenar* disparar, passam a existir dois testemunhos – um em *c1* e outro em *c2* – que representam o mesmo caso. Durante o tratamento de um caso, o número de testemunhos pode flutuar. O número de testemunhos que representam um caso em particular é igual ao número de condições que foram atingidas. Assim que um testemunho se encontra no lugar *fim*, o caso considera-se completo. Em princípio, cada processo deve cumprir dois requisitos: (1) deve ser sempre possível em qualquer momento – através da realização de um certo número de tarefas – atingir um estado em que exista um testemunho no *fim*; e (2) quando existe um testemunho no lugar *fim*, todos os outros devem ter desaparecido da rede. Estes dois requisitos garantem que cada caso que começa no lugar *início* irá ser eventualmente completado de forma correcta. Saliente-se que não é possível ter um testemunho no *fim* enquanto existirem tarefas ainda por realizar. Estes requisitos mínimos, que cada processo deve cumprir, podem ser verificados de

uma forma eficaz utilizando ferramentas *standard* para redes de Petri. O estado de um caso não é determinado unicamente pelas condições que este atingiu; para orientá-lo, o caso poderá possuir um ou mais atributos. Para estes casos parece óbvio que podemos utilizar a extensão por cor das redes de Petri. O valor de um testemunho contém informação sobre os atributos do caso em questão. Iremos aprofundar este assunto em detalhe mais à frente.

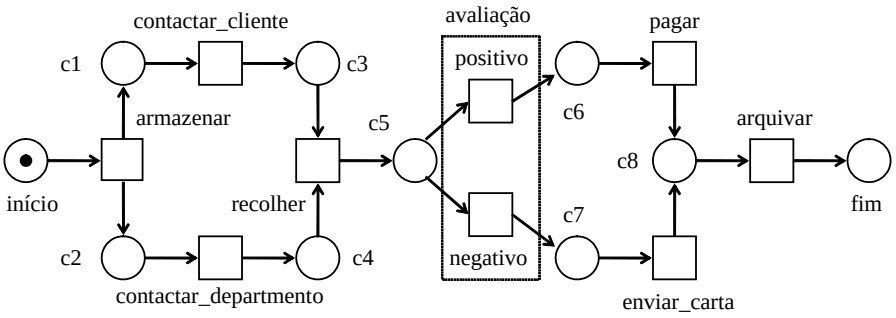


Figura 2.12. O processo "tratar reclamação" modelado com uma rede de Petri

Os testemunhos que correspondem a casos particulares são mantidos estritamente separados (através do sistema de gestão de *workflow*.) Podemos traduzir este pormenor para as redes de Petri utilizando dois métodos. Visto que testemunhos pertencentes a casos diferentes não se podem influenciar um ao outro, podemos produzir uma cópia separada da rede de Petri para cada caso (1º método). Assim, cada um terá o seu próprio processo, tal como ilustrado na figura 2.12. No entanto, podemos também utilizar uma rede de Petri fazendo uso da extensão por cor (2º método). Graças a esta extensão, podemos associar a cada testemunho um valor que identifica o caso ao qual o testemunho pertence. Isto é demonstrado esquematicamente na figura 2.13.

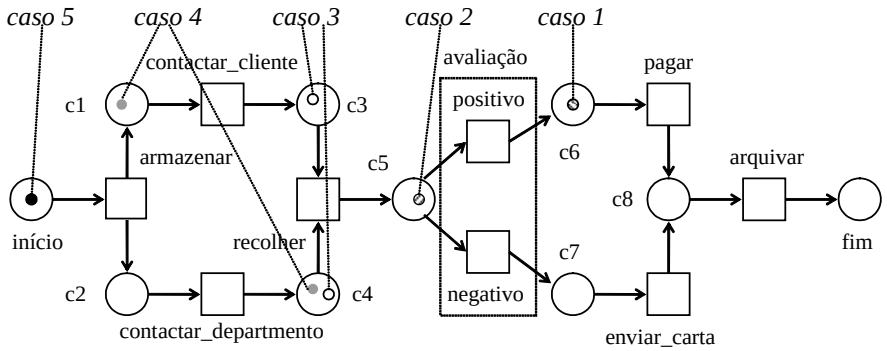


Figura 2.13. Cada caso é ilustrado utilizando um ou mais testemunhos

O estado da rede de Petri ilustrada na figura indica que existem, actualmente, cinco casos em progresso. O caso 1 encontra-se quase completo, ao passo que o

caso 5 ainda está no estado *início*. De forma a assegurar que os testemunhos pertencentes a casos diferentes não são "misturados", cada transição encontra-se provida de uma pré-condição que afirma que somente testemunhos do mesmo caso podem ser consumidos por um disparo. Se a transição *recolher*, ilustrada na figura 2.13, dispara, esta pré-condição vai assegurar que dois testemunhos para o caso 3 são consumidos.

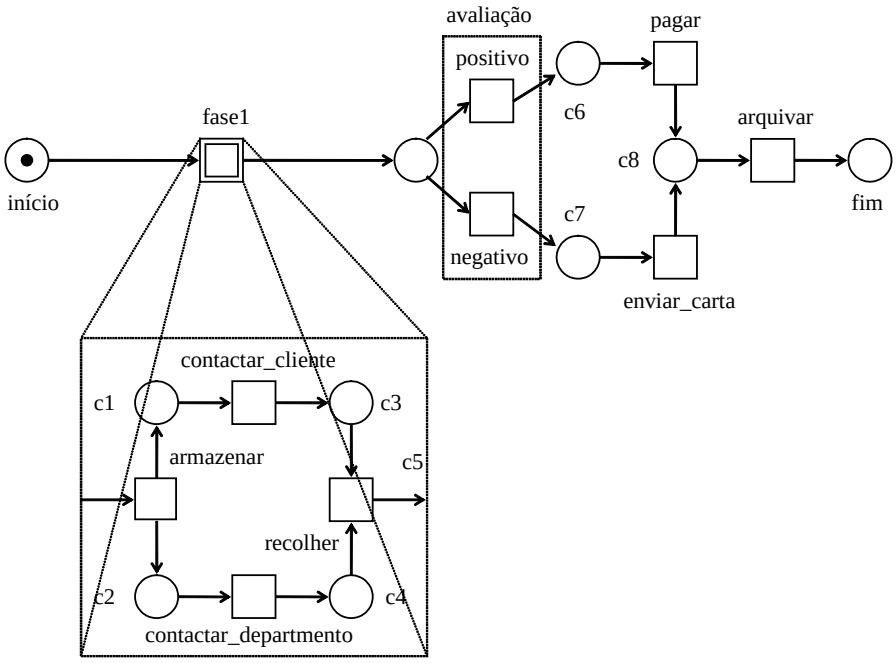


Figura 2.14. O processo "tratar reclamação" contém agora o sub-processo "fase 1"

A figura 2.12 mostra um processo não hierárquico. No entanto, é possível construir o mesmo processo com base em sub-processos. Para ilustrar esta situação, podemos, por exemplo, combinar as primeiras quatro tarefas (*armazenar*, *contactar_cliente*, *contactar_departamento* e *recolher*) num único sub-processo denominado por *fase1*. A figura 2.14 mostra como iria ficar a rede de Petri correspondente, com dois níveis.

2.3.2 Encaminhamento

As tarefas podem ser opcionais. Isto é, podem existir tarefas que só precisam de ser executadas para um certo número particular de casos específicos. A ordem na qual as tarefas são executadas também pode variar de caso para caso. Através do encaminhamento de um caso ao longo de um número de tarefas, podemos determinar que tarefas precisam de ser executadas (e em que ordem devem ser executadas). Como foi referido anteriormente, são reconhecidas quatro construções base para o encaminhamento. Para cada uma delas, iremos ver a rede de Petri correspondente.

(a) **Encaminhamento sequencial**. Referimo-nos à execução sequencial de tarefas quando estas têm que ser executadas uma após a outra. Se duas tarefas precisam de ser executadas sequencialmente, existe normalmente uma clara interdependência entre elas. Por exemplo, o resultado da primeira pode ser necessário para que seja possível executar a segunda. Numa rede de Petri esta forma de encaminhamento é modelada através da ligação das duas tarefas utilizando, para o efeito, um lugar. A figura 2.15 ilustra um exemplo de encaminhamento sequencial.

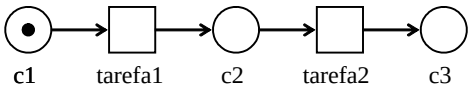


Figura 2.15. Encaminhamento sequencial

A tarefa que corresponde à transição *tarefa2* só é realizada após a tarefa da transição *tarefa1* ter sido completada. Isto é garantido pelo lugar *c2*, que corresponde à condição que tem de ser aplicada antes que a *tarefa2* possa ser executada.

(b) **Encaminhamento paralelo**. Se mais do que uma tarefa pode ser executada ao mesmo tempo ou em qualquer ordem, então referimo-nos ao encaminhamento paralelo. Se nos confinarmos a uma situação com duas tarefas, *tarefa1* e *tarefa2*, então existem três possibilidades de execução: ambas as tarefas podem ser realizadas em simultâneo; a *tarefa1* pode ser executada primeiro, e depois a *tarefa2*; ou a *tarefa2* pode ser executada primeiro, seguida da *tarefa1*. A figura 2.16 ilustra como podemos modelar esta situação utilizando uma rede de Petri.

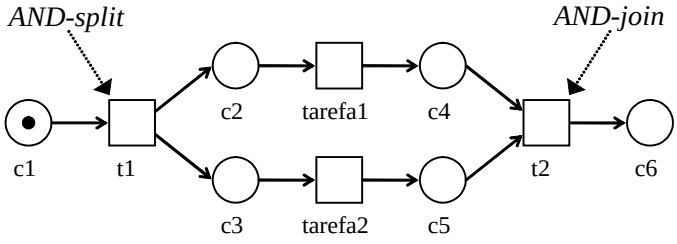


Figura 2.16. Encaminhamento paralelo

De forma a permitirmos a execução paralela das tarefas *tarefa1* e *tarefa2*, no caso correspondente ao testemunho em *c1*, começamos por utilizar aquilo a que chamamos de *AND-split*. Isto é uma tarefa que foi adicionada de forma a permitir que mais do que uma tarefa possa ser gerida ao mesmo tempo. Na figura 2.16, a transição *t1* é o equivalente a um AND-split. Ela dispara quando existe um testemunho em *c1*, e produz um testemunho em *c2* e outro em *c3*. Assim que a condição *c2* é atingida para um caso particular, a *tarefa1* pode ser

executada. Assim que a condição *c3* é atingida, a *tarefa2* também pode ser executada. Logo, o disparar de *t1* permite a realização de duas tarefas. Também dizemos que a *tarefa1* e a *tarefa2* podem ser executadas em paralelo. Somente quando as duas forem realizadas é que a transição *t2* pode disparar. Isto é o equivalente a um *AND-join*: uma tarefa adicionada para sincronizar dois ou mais fluxos paralelos. Somente quando um caso em particular tiver cumprido ambas as condições *c4* e *c5* é que esta tarefa (*t2*) pode ser executada.

Na figura 2.16, tivemos de inserir duas tarefas, *t1* e *t2*, para modelar um AND-split e um AND-join. Chamamos a essas adições "artificiais", *tarefas de gestão*, devido ao facto de elas não corresponderem a uma unidade de trabalho reconhecível ou explícita. Graças a elas, podemos executar a *tarefa1* e a *tarefa2* em paralelo. No entanto, também é possível que tarefas como a *t1* e *t2* correspondam a trabalho útil. Na figura 2.12, por exemplo, a tarefa *armazenar* corresponde a um AND-split. A tarefa *recolher* corresponde a um AND-join.

Num processo de negócio no qual todos os casos são executados de uma forma completamente manual (sem a ajuda de um sistema de *workflow*), o encaminhamento sequencial é normalmente o mais utilizado, devido a limitações físicas ou outras. Por exemplo, as tarefas de um caso específico têm de ser executadas uma após a outra porque o documento associado só pode estar num lugar físico num dado instante. Através da introdução de um sistema de *workflow*, essas limitações são largamente eliminadas. Tarefas que previamente tinham de ser executadas sequencialmente podem agora ser feitas em paralelo. Isto permite, com frequência, obter ganhos consideráveis a nível do tempo de execução dos processos. Por consequência, permitir o encaminhamento paralelo pode influenciar de uma forma significativa o sucesso de um sistema de *workflow*.

(c) **Encaminhamento selectivo**. Um processo determina o encaminhamento para um tipo de caso específico. Mas podem existir diferenças de encaminhamento entre casos individuais. Consideremos, por exemplo, um processo para tratar de participações de seguro. Dependendo das condições específicas de uma participação, um caminho particular terá de ser seleccionado. A tarefa *enviar_analista*, por exemplo, não é executada para pequenas participações. Referimo-nos a tais casos como encaminhamento selectivo. Isto envolve uma escolha entre duas ou mais tarefas. A figura 2.17 ilustra um exemplo modelado utilizando redes de Petri.

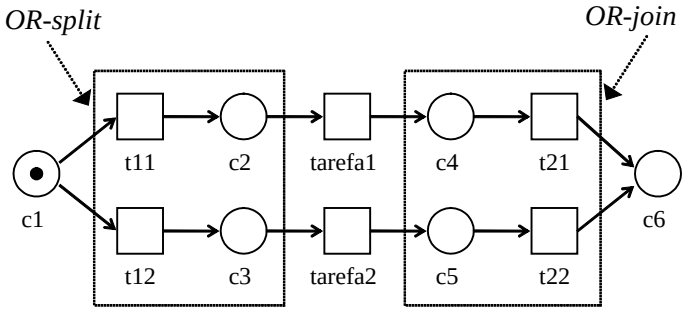


Figura 2.17. Encaminhamento selectivo (1)

Logo que um caso satisfaça a condição *c1*, *t11* ou *t12* dispara. Se for *t11*, então a *tarefa1* é activada, caso contrário, se for *t12*, é a *tarefa2* que é activada. Portanto, existe uma escolha entre as duas tarefas. Designamos a rede que consiste nas transições *t11* e *t12* e nos lugares *c2* e *c3* de um *OR-split*. Assim que uma das tarefas for executada, o *OR-join* garante que um testemunho aparece em *c6*. Neste caso, o OR-join é modelado utilizando uma rede que consiste em 2 lugares (*c4* e *c5*) e duas transições (*t21* e *t22*). Assim sendo, o OR-split selecciona um dos dois caminhos alternativos e o OR-join volta a juntá-los. Na figura 2.17, foram explicitamente modelados o OR-split e o OR-join através da adição de duas pequenas redes. Isto é necessário quando queremos mostrar que o OR-split e o OR-join são tarefas de gestão explícitas. No entanto, é também possível modelá-las implicitamente, como indicado na figura 2.18.

Quando um caso satisfaz a condição *c1*, ou a *tarefa1* ou a *tarefa2* é executada. Isto é mais um exemplo de encaminhamento selectivo. Se olharmos à forma como o OR-join é modelado nas duas figuras anteriores, notamos pouca diferença. Obviamente, então, um OR-join pode ser modelado utilizando diferentes arcos conduzindo ao mesmo lugar. No caso do OR-split, no entanto, já se nota uma diferença. Na figura 2.17, uma escolha é feita no momento em que existe um testemunho em *c1* (isto é, quando um caso cumpre com a condição *c1*). Na figura 2.18, a escolha surge mais tarde. Qual dos dois caminhos é seleccionado só é decidido no momento em que a *tarefa1* ou a *tarefa2* tem que ser executada. Isto pode parecer simplesmente uma diferença subtil, mas na realidade a distinção entre os OR-splits demonstrados nas figuras 2.17 e 2.18 pode ser de uma importância crucial.

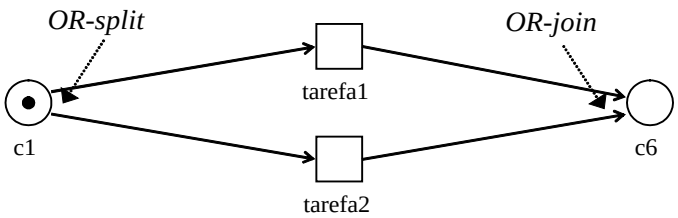


Figura 2.18. Encaminhamento selectivo (2)

Vamos assumir, por exemplo, que a *tarefa1* corresponde ao processamento de um relatório, e que a *tarefa2* deve ser executada caso o relatório não seja entregue dentro de um dado período de tempo. Neste contexto, o modelo fornecido utilizando a construção dada pela figura 2.18 é excelente. Quando o testemunho se encontra em *c1*, dois eventos subsequentes são possíveis: o relatório chega e a *tarefa1* é executada, ou o relatório atrasa-se e a *tarefa2* é então cumprida. A decisão sobre qual a tarefa a executar é atrasada até que o relatório chegue ou até que um dado período de tempo fixo passe. Na figura 2.17, no entanto, a decisão tem de ser tomada imediatamente. Se *t11*, por exemplo, dispara, então deixa de ser possível executar a *tarefa2*. Mais tarde, iremos ver alguns exemplos de maior complexidade, onde o *momento* no qual a escolha é feita é de grande importância.

Até agora, assumimos (implicitamente) que a escolha feita entre as duas alternativas não era determinística. Por outras palavras, ainda não foi explicado como é que a escolha entre a *tarefa1* e a *tarefa2* se processa, devido a – tanto quanto afecta o processo – não ser importante qual das tarefas é realizada: a escolha é deixada ao sistema de *workflow*. Na maioria dos casos, no entanto, uma melhor decisão é tomada de acordo com as propriedades específicas do caso. Dependendo dos valores dos atributos do caso (isto é, os parâmetros de gestão do caso), queremos ser capazes de escolher entre as alternativas. A figura 2.19 mostra como podemos modelar esta situação.

Com base nos atributos do caso, a transição *t1* na figura 2.19 produz um testemunho para *c2* ou para *c3* (mas não para ambos). Neste caso, então, iremos utilizar a extensão por cor das redes de Petri para permitir que uma escolha seja feita na transição *t1*. Utilizando os atributos do caso em questão, a regra de decisão em *t1* determina qual a tarefa a ser executada. Ao fazer isso, assumimos que todos os atributos relevantes para este caso estão contidos dentro do valor do testemunho em *c1*. No caso do encaminhamento paralelo, no entanto, pode existir mais do que um testemunho associado ao mesmo caso. Pelo facto dos atributos estarem associados a todo o caso, estes testemunhos têm de ter valores idênticos. Por outras palavras, nunca devem existir dois testemunhos com valores diferentes associados ao mesmo caso. De forma a garantir isto, temos que assegurar que uma mudança a um atributo de um caso, causada pela execução de uma tarefa, actualiza o valor de todos os testemunhos pertencentes ao caso.

Assim, cada atributo de um caso pode ser visto como informação que pode ser inspeccionada e avaliada por cada tarefa relevante ao caso. Em teoria, a vasta natureza de um atributo de um caso pode ser modelada explicitamente ligando cada transição a um lugar-comum. Este lugar contém sempre um testemunho cujo valor corresponde aqueles existentes nos atributos de um caso. Pelo facto da ilustração deste lugar-comum tornar os diagramas de processos confusos, iremos omiti-los.

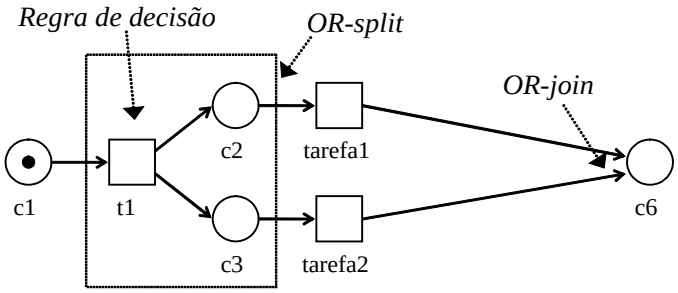


Figura 2.19. Encaminhamento selectivo (3)

Na figura 2.19, o número de testemunhos produzidos em cada um dos lugares de saída de *t1* é variável (0 ou 1). Uma escolha é feita baseada somente no valor (atributos do caso) do testemunho em *c1* e na regra de decisão em *t1*. No entanto, podemos também exprimir esta escolha utilizando duas transições contendo as pré-condições adequadas. É de recordar que uma pré-condição é

baseada nas cores dos testemunhos a serem consumidos e actua como um guarda a transição. A figura 2.20 mostra como é que isto é possível.

A pré-condição na transição *t11* corresponde aos requisitos que necessitam de ser satisfeitos de forma a justificar a escolha da *tarefa1*. A pré-condição em *t12* determina quando é que a *tarefa2* deve ser seleccionada. Se a pré-condição em *t11* é a negação da pré-condição em *t12*, então cada testemunho em *c1* irá resultar numa escolha determinística entre a *tarefa1* e a *tarefa2*. Neste caso os OR-splits nas figuras 2.19 e 2.20 são equivalentes.

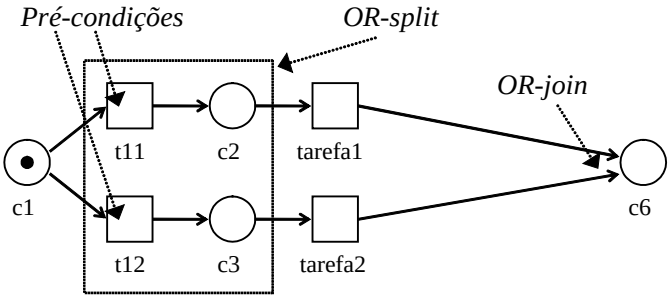


Figura 2.20. Encaminhamento selectivo (4)

Porque construções tais como o AND-split, o AND-join, o OR-split e o OR-join ocorrem com frequência, utilizamos uma notação especial para representá-los. Isto é ilustrado na figura 2.21.

Representamos um AND-split utilizando o símbolo ⊠ no lado da saída. Isto indica que um testemunho deve ser produzido por cada um dos lugares de saída em todas as circunstâncias.

Representamos um AND-join utilizando o símbolo ⊠ no lado de entrada. Isto indica que a tarefa a ser modelada só pode ser executada quando existe um testemunho em cada um dos lugares de entrada. Com base na figura 2.21, podemos ver que ambos os AND-split e AND-join correspondem a uma "transição normal", tal como aquelas encontradas na rede de Petri clássica.

Devemos representar um OR-split através da utilização do símbolo ⊠ no lado de saída. Este indica que um testemunho deve ser produzido para precisamente um dos lugares de saída. Como vimos anteriormente, podemos modelar esta situação de duas formas. Nas seguintes secções deste capítulo, iremos usar unicamente a primeira forma.

Representamos um OR-join usando o símbolo ⊠ no lado de entrada.

Podemos utilizar a seguinte técnica para relembrar a diferença entre os símbolos do AND e do OR. Quando os arcos entram ou saem do mesmo triângulo, é um AND. Caso contrário é um OR.

O símbolo ⊟ no lado da saída indica uma mistura de um AND-split e de um OR-split. Neste caso, um ou mais testemunhos serão produzidos, dependendo dos valores dos atributos do caso. A figura 2.21 mostra duas formas de usar esta estrutura híbrida numa rede de Petri.

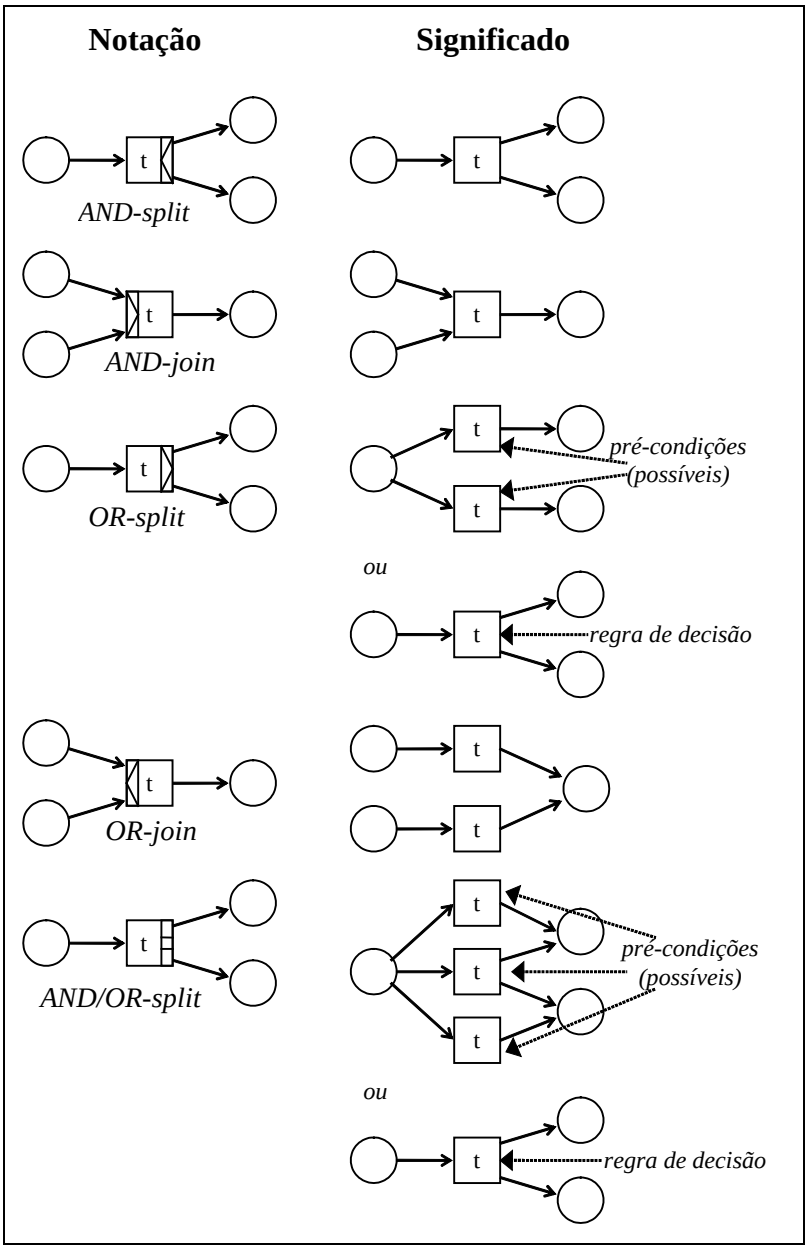


Figura 2.21. Notação utilizada para construções comuns

(d) **Encaminhamento iterativo**. A última forma de encaminhamento é a execução repetitiva de uma tarefa. Idealmente, uma tarefa irá ser executada uma única vez por caso. Em certas situações, no entanto, é necessário aplicar o *encaminhamento iterativo*. Por exemplo, quando uma dada tarefa precisa de ser repetida até que os resultados de um teste subsequente provem ser positivos. A figura 2.22 mostra como podemos modelar o encaminhamento iterativo.

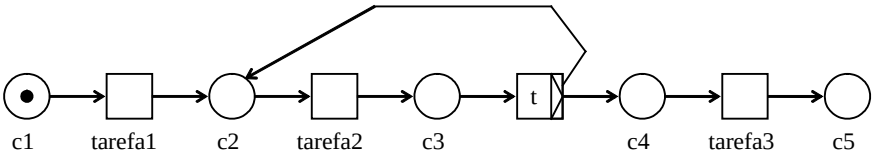


Figura 2.22. Encaminhamento iterativo (1)

Considerando o caso correspondente ao testemunho em *c1*, verificamos que a *tarefa1* e a *tarefa2* são executadas sucessivamente. Assim que a *tarefa2* completar a sua execução, o OR-split *t* determina se deve ou não ser executada novamente. Assim que a *tarefa2* tiver sido executada uma ou mais vezes, o caso passa para a *tarefa3*. A *tarefa2* deve ser executada pelo menos uma vez entre a *tarefa1* e a *tarefa3*.

A figura 2.22 assume que a *tarefa2* deve ser executada pelo menos uma vez ("repeat … until … "). Se não for o caso, aplica-se a construção ilustrada na figura 2.23 ("while … do …").

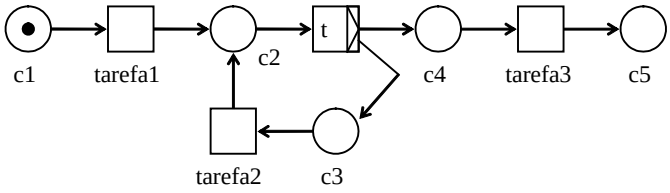


Figura 2.23. Encaminhamento iterativo (2)

Imediatamente após a conclusão da *tarefa1*, o OR-split *t* determina se a *tarefa2* necessita ou não de ser executada. Torna-se agora possível para a *tarefa1* ser seguida de imediato pela *tarefa3*.

Em ambos os exemplos, existe um OR-split que toma a sua decisão com base nos valores dos atributos do caso. De notar que as duas construções ilustradas correspondem às construções já familiares "repeat … until …" e "while … do …" que aparecem em muitas linguagens de programação.

Exemplo: Utilizando o exemplo descrito no capítulo anterior, podemos agora ilustrar os conceitos definidos até ao momento. O exemplo refere-se a uma companhia de seguros e ao processo existente na mesma para lidar com participações. O capítulo 1 identifica 16 tarefas neste processo. No entanto, ainda não tinha sido introduzido o conceito das redes de Petri para modelar processos de *workflow*. Devido a esse facto, tínhamos recorrido a uma técnica de notação "ad-hoc" para ilustrar o encaminhamento. Agora, contudo, já podemos mostrar o processo de uma forma "correcta", tal como ilustrado na figura 2.24. Mas antes de olhar para o diagrama, tente modelar o processo.

De forma a melhor entendermos o diagrama, as condições que são utilizadas para encaminhar os casos correctamente "receberam" nomes "simbólicos".

Contudo, nomes simbólicos não têm utilização na prática. Por exemplo, poderíamos mais correctamente denominar a condição *c7* de *aceitação*. As condições *c1* e *c20* têm um papel especial: *c1* representa o início do processo e *c20* o seu fim. Note-se que o diagrama "informal" do capítulo 1 e a figura 2.24 possuem semelhanças. A diferença que mais se destaca entre os dois é que as figuras se encontram identificadas de uma forma explícita na figura 2.24. Como resultado, podemos descrever o estado de um caso.

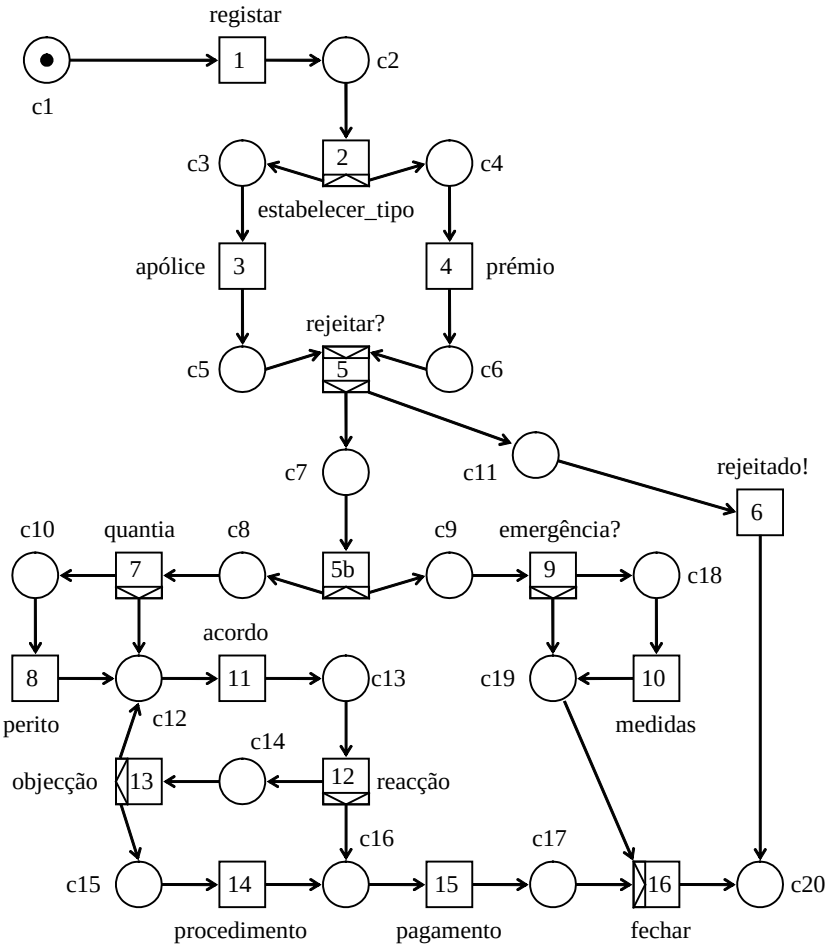


Figura 2.24. O processo para tratar de participações de seguro

2.3.3 Execução

Um processo é uma colecção de tarefas, condições, sub-processos e as suas relações. Como vimos anteriormente, podemos descrever um processo usando uma rede de Petri. As condições são representadas usando lugares e as tarefas

usando transições. De maneira a simplificar a representação de um processo em termos de uma rede de Petri, definimos um método para descrever as construções típicas. (ver figura 2.21).

Um processo é concebido para tratar de uma *categoria* particular de casos, e, por isso, pode gerir vários casos individuais. Uma tarefa não é específica a um caso particular. No entanto, quando um caso está a ser tratado por um processo, as tarefas são executadas para esse caso específico. De forma a evitar uma certa confusão entre a tarefa em si e a sua execução no contexto de um caso específico, introduzimos os termos *item de trabalho* e *actividade*. Um item de trabalho é a combinação de um caso com uma tarefa que está pronta a ser executada. O termo actividade refere-se a execução do item de trabalho. Quando um item de trabalho está a ser executado é transformado numa actividade. Note que ao contrário da tarefa, ambos o item de trabalho e a actividade estão ligados a um caso específico. A distinção entre (1) uma tarefa, (2) um item de trabalho, e (3) uma actividade tornar-se-á clara assim que forem mapeadas em termos de redes de Petri. Uma tarefa corresponde a uma ou mais transições, um item de trabalho a uma transição pronta, e uma actividade ao disparo de uma transição.

As transições numa rede Petri são classificadas de "ansiosas". Por outras palavras, elas disparam logo que estejam prontas. Como acabámos de estabelecer, uma transição pronta corresponde a um item de trabalho. No entanto, executar um item de trabalho requer muitas vezes mais do que um caso estar no estado apropriado. Se for levado a cabo por uma pessoa, ele deve primeiro retirá-lo da sua "lista de tarefas" antes da actividade poder começar. Por outras palavras, o item de trabalho apenas é executado quando o funcionário toma a iniciativa. É por estes casos que se reconhece a existência de mecanismos de disparo. Certos items de trabalho só podem ser transformados numa actividade após terem sido disparados.

Diferenciamos três tipos de *disparos*: (1) por iniciativa de um recurso (tal como um funcionário retirar um item de trabalho da sua "lista de tarefas"); (2) um evento externo (tal como a chegada de uma mensagem EDI); e (3) um sinal temporal (tal como a criação de uma lista de requisições às seis horas em ponto). Os items de trabalho que têm de ser executados imediatamente, sem a intervenção de um recurso, não precisam de um disparo. Podemos ilustrar usando uma rede de Petri de que forma um disparo é aplicado. As tarefas disparadas por um recurso são ilustradas usando uma seta larga apontando para baixo ( ). Aquelas que são disparadas por um evento externo possuem um símbolo representado por um envelope ( ). Aquelas que são dependentes do tempo possuem um símbolo de relógio ( ). A figura 2.25 mostra um exemplo de um processo contendo informação sobre os disparos.

A *tarefa2* e a *tarefa4* são geridas por um recurso. A *tarefa3* é dependente do tempo e a *tarefa1* requer um disparo externo (por exemplo, uma mensagem EDI). A única tarefa automática é a *tarefa5*.

A noção do mecanismo de disparo é de grande importância. Não é o sistema de *workflow* que é responsável por esses eventos mas sim o ambiente. O sistema não pode forçar um cliente a responder a um pedido; nem mesmo forçar um funcionário a executar um item de trabalho num determinado instante. É fácil modelar o mecanismo de disparo em termos de uma rede de Petri. Para cada transição pertencente a uma tarefa que requer um disparo, um lugar de entrada extra é adicionado. Um testemunho nesse lugar de entrada

extra representa um disparo. Portanto, quando um disparo é registado pelo sistema de *workflow*, aparece um testemunho na entrada extra.

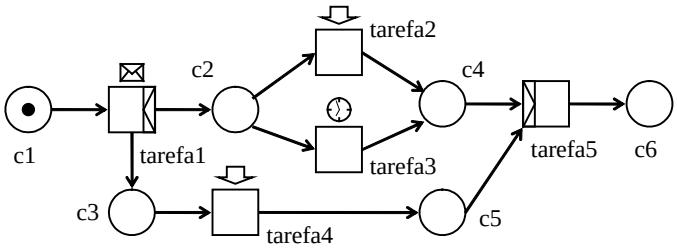


Figura 2.25. Um exemplo com vários tipos de disparos

O mecanismo de disparo também mostra que a *temporização de uma escolha OR-split* é crucial. Na figura 2.25, a temporização de uma escolha não determinista entre a *tarefa2* e a *tarefa3* é o mais tarde possível. Quando se chega a condição *c2*, existem duas possibilidades. A primeira é que o funcionário inicia o item de trabalho correspondente à *tarefa2* antes do momento especificado para o desempenho da *tarefa3*. Alternativamente, nenhum funcionário toma a iniciativa de executar a *tarefa2* antes desse momento. No primeiro caso, a *tarefa2* dispara, no segundo, a *tarefa3* dispara. Uma escolha entre as duas alternativas é então atrasada até ao momento em que um primeiro disparo seja recebido. Como não sabemos de antemão qual dos disparos será activado, o OR-split implícito no lugar *c2* não pode ser substituído por um OR-split explícito na forma de uma ou duas transições adicionais. Portanto o OR-split pode ter duas formas: *implícita* e *explícita*. A figura 2.26 mostra estas formas graficamente.

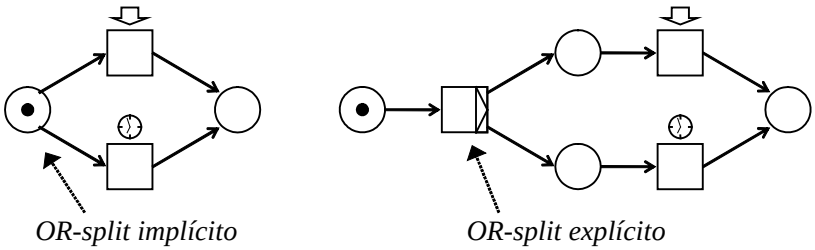


Figura 2.26. Existe uma diferença essencial entre o OR-split implícito e explícito

Tal como no disparo de uma transição, uma actividade – isto é, a execução concreta de uma tarefa para um caso específico – é uma unidade atómica, por isso é sempre executada na totalidade. No entanto, pode ocorrer uma falha durante o desempenho de uma tarefa relacionada com a actividade. Por exemplo, pode fazer uso de um recurso (tal como um funcionário) que a interrompe por alguma razão. Um funcionário pode reconhecer que certos

dados necessários para executar a tarefa estão em falta, ou então a actividade pode usar uma aplicação (tal como um programa para calcular as taxas de juro) que bloqueia durante a execução da tarefa. Além do mais, uma falha no sistema de *workflow* – talvez devido a um erro do sistema – durante uma actividade não pode ser ignorada.

Em todos estes casos, é necessário um *rollback*. Isto envolve colocar o sistema de *workflow* no seu estado anterior antes do início da actividade que não foi executada com sucesso. Após este *rollback*, a actividade pode ser reiniciada. Apenas quando a actividade for completada com sucesso, é que ocorre um *commit* e todas as mudanças efectuadas tornam-se definitivas. Em relação ao processo, um *rollback* é muito simples: os atributos de um caso e todas as condições validas são repostas com os seus valores iniciais. Para a aplicação (que foi interrompida a meio do desempenho de uma tarefa), um *rollback* pode ser mais complicado.

2.3.4 Exemplo: Agência de Viagens

Vamos considerar um exemplo onde os disparos têm um papel importante. Para organizar uma viagem, uma agência de viagens executa várias tarefas. Em primeiro lugar, o cliente é registado. De seguida, o funcionário procura alternativas que são comunicadas ao cliente. Depois, o cliente será contactado para aferir se ele ou ela ainda está interessado na viagem desta agência e se são desejadas mais alternativas. Existem três possibilidades: (1) o cliente não está interessado, (2) o cliente gostaria de ver mais alternativas e (3) o cliente selecciona uma alternativa. Se o cliente selecciona uma viagem, a viagem é reservada. Em paralelo, um ou dois tipos de seguro são preparados se forem desejados. Um cliente pode escolher um ou mais seguros para cancelar a viagem e/ou para salvaguardar a possível perda de bagagem. Note que o cliente pode optar por: (1) não possuir qualquer seguro, (2) optar apenas pelo seguro de cancelamento de viagem, (3) adquirir apenas o seguro de perda de bagagem ou (4) subscrever ambos os tipos de seguro. Duas semanas antes da data de partida, os documentos são enviados ao cliente. Uma viagem pode ser cancelada a qualquer momento depois de completo o processo de reserva (incluindo o seguro) e antes da data de partida. Note que os clientes que não possuem seguro para o cancelamento da viagem podem cancelar a viagem (mas não serão reembolsados).

Com base nesta descrição informal, criámos o processo correspondente usando as construções introduzidas neste capítulo. A figura 2.27 mostra o resultado.

O processo, tal como em qualquer outro processo de *workflow* deste livro, possui um lugar de origem que serve de condição inicial (i.e., a criação de casos) e um lugar de destino que representa a condição final (i.e., a finalização de casos). Primeiro, as tarefas *registar*, *procurar*, *comunicar* e *contactar_cliente* são executadas sequencialmente. A tarefa *contactar_cliente* é um OR-split com três saídas possíveis: (1) o cliente não está interessado, isto é, um testemunho é colocado no lugar *fim*, (2) o cliente gostaria de ver mais alternativas, isto é, um testemunho é colocado em *c2*, e (3) o cliente selecciona uma alternativa, isto é, um testemunho é colocado em *c15* para iniciar a reserva. As tarefas *AND_split* e *AND_join* foram adicionadas apenas para propósitos de encaminhamento. Estas

tarefas de encaminhamento possibilitam a execução paralela das tarefas de reserva e de seguro. A tarefa *reservar* corresponde à reserva da viagem.

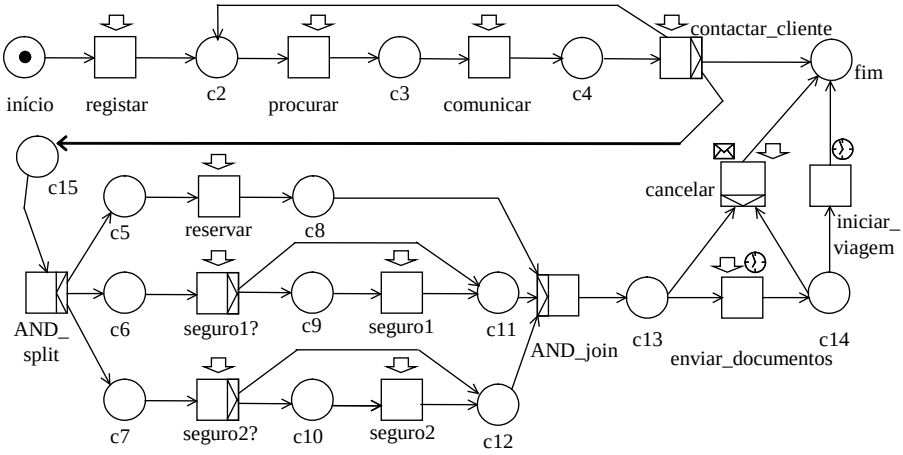


Figura 2.27. A agência de viagens

As tarefas *seguro1* e *seguro2* correspondem à gestão de ambos os tipos de seguros. Uma vez que ambos os tipos de seguro são opcionais, existe um *bypass* para cada uma destas tarefas. O OR-split *seguro1?* permite um *bypass* da tarefa *seguro1* colocando um testemunho em *c11*. Após manusear a reserva e os seguros opcionais, o AND-join coloca um testemunho em c13. O resto do processo é, do ponto de vista dos disparos, muito interessante. Note que todas as tarefas executadas antes deste ponto, ou são tarefas que requerem um disparo de recurso, ou são tarefas automáticas adicionadas apenas com o propósito de encaminhamento. As setas que apontam para baixo representam os disparos de recursos. Se o caso está no lugar *c13* então o fluxo normal da execução é de primeiro executar a tarefa *enviar_documentos* e depois executar *iniciar_viagem*. Note que a tarefa *enviar_documentos* requer tanto um disparo de recurso como um disparo de tempo. Estes dois disparos indicam que duas semanas antes do início da viagem, um funcionário envia os documentos ao cliente. A tarefa *iniciar_viagem* foi adicionada com o propósito de encaminhamento e requer um disparo de tempo. Sem a tarefa *iniciar_viagem*, isto é, colocando o testemunho no lugar *fim* depois de enviar os documentos, seria impossível cancelar a viagem após ter enviado os documentos. A tarefa *cancelar* é um OR-join explícito e requer tanto um disparo de recurso como um disparo externo. Esta tarefa é executada apenas se for disparada pelo cliente. A tarefa *cancelar* só pode ser executada quando o caso está em *c13* ou *c14*, isto é, após gerir as tarefas relacionadas com a reserva e os seguros e antes do início da viagem.

Usando a agência de viagens como exemplo, podemos apontar *duas linhas orientadoras para a modelação*. A primeira linha orientadora trata do uso de OR-joins. *As tarefas OR-join devem ser evitadas tanto quanto possível*. Na maior parte das situações é possível usar lugares/condições em vez de modelar tarefas OR-join explicitamente. Se uma tarefa OR-join possui duas ou mais condições de entrada e estas condições não são entradas para nenhuma outra tarefa, então

estas condições podem ser fundidas porque, de um ponto de vista semântico, elas são idênticas. Como resultado, o número de elementos no diagrama é reduzido e não há necessidade de usar um OR-join. Por exemplo, o lugar *c2* na figura 2.27 pode ser dividido em duas condições; uma condição para novos casos e uma condição para casos que requerem mais trabalho. Tal divisão introduziria a necessidade de da tarefa OR-join *procurar*. O diagrama resultante torna-se mais complexo sem mudar o comportamento do processo. Desta forma, preferimos a solução com uma condição *c2* com dois arcos de chegada. Apenas em raras situações, as tarefas OR-join são realmente necessárias para obter o comportamento desejado. Considere por exemplo a figura 2.27. A tarefa *cancelar* é um OR-join. Não é possível remover este OR-join fundindo as condições de entrada *c13* e *c14*. As condições *c13* e *c14* correspondem a estados diferentes, isto é, em *c13*, *enviar_documentos* está pronta e em *c14* *iniciar_viagem* está pronta. A segunda linha geral para modelação trata do *uso de disparos para a primeira tarefa* do processo. Na figura 2.27 poderíamos ter adicionado um disparo externo à tarefa *registar*. Este disparo iria corresponder ao pedido do cliente. Outra interpretação é que o pedido do cliente corresponde à criação do testemunho inicial no lugar *iniciar*. Esta interpretação é usada na figura 2.27. Desta forma, um disparo externo não foi adicionado à tarefa *registar*. Neste livro, preferimos usar esta interpretação. No entanto, a interpretação em que a primeira tarefa requer um disparo externo para iniciar o processo também é permitida.

Para finalizar… Neste capítulo, introduzimos uma técnica de modelação de processos para a especificação de *workflows*. Esta é baseada na teoria de redes de Petri e possui um conjunto de vantagens. Primeiro, a técnica é gráfica e fácil de aplicar. Como constatamos usando vários exemplos, os conceitos de *workflow* podem ser ilustrados com elegância usando redes de Petri. Segundo, é uma técnica com um bom fundamento formal: o significado de cada processo é definido com precisão. Como resultado, descobrimos, por exemplo, que existem dois tipos de OR-split. Outra vantagem importante sobre muitas outras técnicas de modelação de processos é o facto de (temporariamente) os estados serem indicados explicitamente. Isto possibilita diferenciar entre um OR-split implícito e explícito. Os estados explícitos tornam conceptualmente mais fácil cancelar casos. O cancelamento pode ser alcançado simplesmente removendo todos os testemunhos que pertencem a esse caso. Uma noção explícita de estados é também essencial quando se transfere um caso de um sistema de *workflow* para outro. Finalmente – devido às bases formais das redes de Petri – existem vários métodos analíticos disponíveis.

EXERCÍCIOS

Exercícios de redes de Petri clássicas

Exercício 2.1 Semáforo Alemão

Existem algumas diferenças entre os semáforos nos diferentes países. Os semáforos descritos neste capítulo são germânicos. Os semáforos na Alemanha possuem uma fase extra no seu ciclo. Os semáforos alemães não passam subitamente do vermelho para o verde, mas passam pelo amarelo antes de chegar ao verde.

(a) Identifique os possíveis estados e modele o sistema de transições. Um sistema de transições apresenta uma lista com todos os estados possíveis e as transições entre os estados.
(b) Desenhe uma rede de Petri que seja capaz de se comportar como um semáforo alemão. Devem existir três lugares que representam o estado de cada luz e todas as transições entre os estados do sistema de transições devem ser suportados.
(c) Elabore uma rede de Petri que se comporte exactamente como um semáforo alemão. Certifique-se que a rede de Petri não permite transições entre estados que não sejam possíveis.

Exercício 2.2 O projecto X

Um projecto secreto do governo (vamos chamá-lo Projecto X) será executado por uma pessoa e consiste em 6 tarefas: A, B, C, D, E e F. A figura 2.28 especifica a ordem pela qual as tarefas têm de ser executadas (gráfico de precedência, cf. PERT/CPM). Uma execução possível é, por exemplo, ABCDEF.

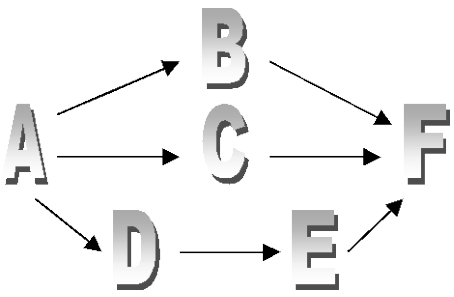


Figura 2.28. O projecto X

(a) Modele o projecto em termos de uma rede de Petri clássica.
(b) Como desenharia o modelo para que E fosse opcional?
(c) Como desenharia o modelo para que as tarefas D e E fossem executadas consecutivamente, isto é, B e C não são permitidas entre D e E?

Exercício 2.3 Caminhos-de-ferro

Uma rede circular de caminhos-de-ferro é composta por quatro linhas. Cada linha pode estar num dos três estados seguintes:

- Ocupada, isto é, há um comboio na linha.
- Requisitada, ou seja, um comboio requisitou com sucesso o acesso à linha.
- Livre, isto é, nem ocupada nem reservada.

Existem dois comboios usando a linha circular. A linha na qual está um comboio está ocupada. Para mudar para a próxima linha, um comboio primeiro requisita a próxima linha. Apenas as linhas livres podem ser requisitadas. As linhas ocupadas são libertas no momento em que o comboio muda de linha. Podemos abstrair-nos da identidade dos comboios, apenas o estado da rede de caminhos-de-ferro é considerado.

(a) Modele o comportamento dinâmico da rede de caminhos-de-ferro usando uma rede de Petri.
(b) Seria fácil modelar a situação com dez linhas (160 estados!)?

Exercício 2.4 Contador binário

O seguinte contador (binário) é para ser modelado com uma rede de Petri. A presença ou ausência de um testemunho num lugar representa um valor binário (1 ou 0). A combinação das marcas destes lugares representa um número natural, que é mostrado pelo contador. Por exemplo, o número binário 101, ou seja, 5, marca dois lugares correspondentes a um "1" (i.e., os lugares 2^2 e 2^0) e um lugar correspondente a um "0" (i.e., o lugar 2^1).

Faça um modelo de um contador capaz de contar de 0 a 7.

Exercícios de redes de Petri de alto nível

Exercício 2.5 Escola de condução

Uma escola de condução está a tentar criar um sistema de informação para monitorizar o progresso do treino dos seus estudantes e o desempenho dos instrutores. Como ponto de partida para a criação de um processo formal, podemos usar a seguinte descrição.

Os novos estudantes registam-se na escola de condução. Um estudante registado tem uma ou mais lições de condução seguidas de um exame. Cada lição de condução possui um início e um fim. Os instrutores dão lições de condução. A escola de condução possui cinco instrutores. Cada lição de condução é seguida, ou por outra lição, ou por um exame. O exame possui um início e um fim e é supervisionado por um examinador. No total existem dez examinadores. Existem três possibilidades para o resultado de um exame:

1. O estudante passa e sai da escola de condução.
2. O estudante reprova e tem mais aulas de condução com o objectivo de tentar novamente.
3. O estudante reprova e desiste.

(a) Modele a escola de condução em termos de uma rede de Petri clássica.
(b) Use uma rede de Petri colorida para modelar que uma pessoa tem dez lições antes de um exame e será excluída se reprovar três vezes.
(c) Adicione tempo para modelar que uma lição demora uma hora e um exame 30 minutos.

Exercício 2.6 Fábrica de bicicletas

Uma fábrica produz bicicletas (apenas de um tipo). A lista de material (*bill-of-materials*) é apresentada na figura 2.29.

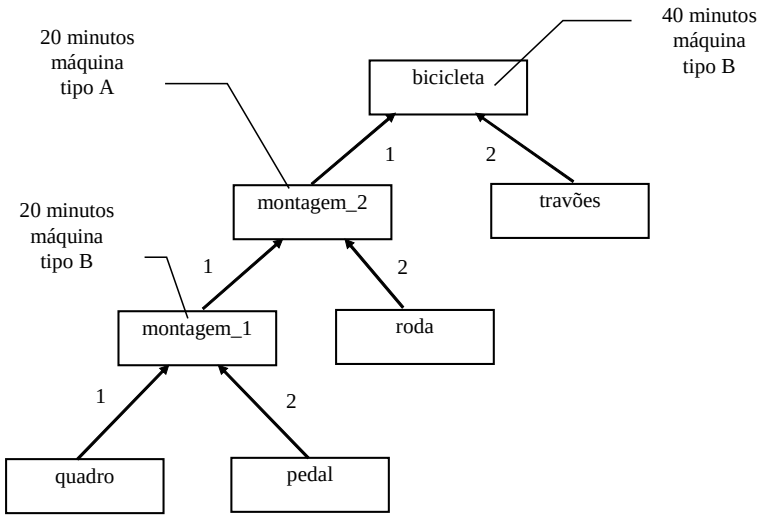


Figura 2.29. A fábrica de bicicletas

Os fornecedores entregam a matéria-prima. Primeiro, o quadro e os dois pedais são montados. Isto demora 20 minutos e é feito por uma máquina do

tipo B. Os outros dois passos da montagem são definidos de um modo semelhante (ver figura 2.29). De seguida, o produto final é entregue após 3 passos de montagem. A fábrica possui três máquinas do tipo A e sete máquinas do tipo B. Cada máquina tem capacidade 1, isto é, a máquina ou está livre ou está ocupada.

(a) Modele a fábrica em termos de uma rede de Petri. Certifique-se que modela os estados das máquinas (ocupado/livre) explicitamente abstraindo-se do tempo. (b) Adicione tempo para modelar o comportamento temporal. Qual é a taxa de produção máxima por hora?

Definições de Processos de Workflow

Exercício 2.7 Companhia de seguros

A companhia de seguros X processa formulários resultantes de acidentes rodoviários com carros em que os seus clientes estiveram envolvidos. Para tal, usa o seguinte procedimento para processar as participações de seguro.

Cada formulário, reportado por um cliente, é registado por um funcionário do departamento DA (Danos Automóveis). Após o registo do formulário, o formulário de seguro é classificado por um gestor do tipo A ou B no departamento DA. Existem duas categorias: formulários simples e complexos. Para formulários simples, é necessário executar duas tarefas: verificar o seguro e telefonar para o mecânico. Estas tarefas são independentes uma da outra. Os formulários complexos requerem a execução de três tarefas: verificar o seguro, verificar o histórico de danos e telefonar para o mecânico. Estas tarefas necessitam de ser executadas sequencialmente na ordem especificada. Para ambos os formulários, simples e complexos, as tarefas são desempenhadas por empregados do departamento DA. Após executar as duas ou três tarefas, uma decisão é tomada. A decisão é tomada por um gestor do tipo A e tem duas possibilidades de resposta: OK (positivo) e NOK (negativo). Se a decisão for positiva, então a companhia de seguros X tem de pagar. Um funcionário do departamento de finanças trata do pagamento. Em qualquer dos casos, a companhia de seguros envia uma carta ao cliente que preencheu o formulário. O funcionário do departamento DA escreve a carta.

Modele o *workflow* fazendo uma definição do processo em termos de uma rede de Petri, usando as técnicas introduzidas neste capítulo.

Exercício 2.8 Gestão de reclamações

Todos os anos, a agência de viagens Y tem de processar muitas reclamações (cerca de 10000). Há um departamento especial para processar estas reclamações (o departamento R). Existe também um departamento interno chamado logística (o departamento L), que trata do registo das reclamações que chegam e do arquivo das reclamações processadas. O procedimento seguinte é usado para manusear estas reclamações.

Primeiro, um funcionário do departamento L regista todas as reclamações que chegam. Após o registo, um formulário é enviado ao cliente com questões acerca da natureza da reclamação. Isto é feito por um funcionário do departamento R. Existem duas possibilidades: o cliente devolve o formulário dentro de duas semanas ou não. Se o formulário for devolvido, é processado automaticamente, resultando num relatório que pode ser usado para realmente processar a reclamação. Se o formulário não for devolvido a tempo, ocorre um *time-out* resultando num relatório vazio. Note que isto não quer dizer necessariamente que a reclamação foi descartada. Após o registo, isto é, em paralelo com o manuseamento do formulário é iniciada a preparação para o verdadeiro processamento.

Primeiro, a reclamação é avaliada por um gestor de reclamações do departamento R. A avaliação mostra que é necessário um processamento mais alongado ou não. Note que esta decisão não depende do manuseamento do formulário. Se não for necessário um processamento mais aprofundado e se o formulário já foi manuseado, a reclamação é arquivada. Se for necessário um processamento mais aprofundado, um funcionário do departamento R executa a tarefa "processar reclamação" (ou seja, o processamento propriamente dito onde certas acções são propostas se for necessário). Para o processamento propriamente dito da reclamação, é usado o relatório resultante do manuseamento do formulário. Note que o relatório pode estar vazio. O resultado da tarefa "processar reclamação" é verificado por um gestor de reclamações. Se o resultado não é OK, a tarefa "processar reclamação" é executada novamente. Isto é repetido até que o resultado seja aceitável. Se o resultado é aceite, um funcionário do departamento R executa as acções propostas. Seguidamente, a reclamação processada é arquivada por um funcionário do departamento L.

Faça o processo, ou seja, modele o *workflow* fazendo uma definição de processo em termos de uma rede de Petri.

Exercício 2.9 Vamos dar uma festa

Um grupo de estudantes quer estabelecer uma agência para organizar festas. O cliente deve indicar o montante a ser gasto, o número de pessoas que se espera na festa e a área na qual a festa deve ser dada. Com esta informação, a agência procura uma localização adequada e toma conta do resto.

As localizações são do tipo *indoor* (interior) ou *outdoor* (exterior). Se a localização for *indoor*, uma sala é alugada. No entanto, no caso de uma localização *outdoor*, uma tenda para festividades e um terreno têm de ser disponibilizados, possivelmente com uma permissão para fazer barulho (música). Há dois tipos de música: ao vivo ou CDs. A escolha entre estas alternativas, não é feita pelo cliente, mas sim pela agência: a música ao vivo é preferível, mas dispendiosa, portanto, a maior parte das festas terá música de CDs. Os CDs são também escolhidos se não houver tempo suficiente para contratar uma banda. Se os CDs forem escolhidos, um sistema de som terá de ser disponibilizado. No caso da música ao vivo, as coisas são mais complicadas. Primeiro, a banda é seleccionada. De seguida, é enviada uma carta a convidar a banda a tocar na festa. Se a banda não reagir dentro de uma semana, um nova banda é seleccionada e o procedimento é repetido. Se reagirem, existem novamente duas possibilidades: estão interessados ou não estão interessados. No

último caso, uma nova banda é seleccionada e o procedimento é repetido. No entanto, no primeiro caso, a banda não é contratada imediatamente. Primeiro, a agência deve ver e ouvir a banda para ver se são suficientemente bons. Devido ao facto dos estudantes apenas quererem o melhor, só cerca de trinta por cento das bandas são consideradas suficientemente boas. Para os outros setenta por cento, uma nova banda é seleccionada, e assim por diante. Se os estudantes não conseguirem encontrar uma banda a tempo, eles usam CDs. É claro que as bandas que tenham sido contratadas anteriormente não necessitam de ser avaliadas. São contratadas imediatamente. Depois de tratar da localização e da música, eles também tratam da comida e das bebidas. No caso de se contratar uma banda, tem de se comprar comida e bebidas extra para os músicos. Para se certificarem que tudo está bem, os estudantes dão uma olhada à festa quando esta está a ser dada. Depois, a conta é enviada ao cliente.

(a) Modele o *workflow* fazendo uma definição do processo em termos de uma rede de Petri, usando as técnicas introduzidas neste capítulo. Atribua disparos às tarefas sempre que achar apropriado.
(b) Analise o processo e investigue possíveis melhoramentos.

(Página deixada propositadamente em branco)

Capítulo 3.

GESTÃO DE WORKFLOWS

3.1 Conceitos de Gestão de Recursos

Usando a definição de processo, podemos indicar *quais* as tarefas que precisam de ser efectuadas para uma categoria particular de casos. Podemos também mostrar a ordem pela qual elas têm de ser efectuadas. Contudo, a definição do processo não indica *quem* deve executar as tarefas. Mas a forma como os itens de trabalho são associados aos recursos (pessoas e/ou máquinas) é muito importante para a eficiência e eficácia dos *workflows*. Neste capítulo, iremos concentrarmo-nos na gestão dos recursos e na ligação entre a definição de um processo e os recursos disponíveis. Iremos também prestar atenção em como melhorar *workflows*.

3.1.1 O recurso

Um sistema de *workflow* tem como objectivo suportar processos de negócio. Num processo, o trabalho é executado por meios de produção, também conhecidos por *recursos*. Num ambiente administrativo, o termo recurso refere-se primariamente ao pessoal de escritório. No entanto, um doutor, uma impressora, um porteiro e um robot que opera na linha de montagem são exemplos de recursos. A característica básica de um recurso é que pode executar tarefas específicas. Também assumimos que cada recurso é identificável univocamente e que tem uma certa capacidade. Neste capítulo, iremos confinarmo-nos a recursos de capacidade igual a um. Ou seja, cada recurso só pode estar a trabalhar em apenas uma actividade num dado instante. Contudo, isto não tem de ser o caso na prática.

3.1.2 Classificação de recursos

Em geral, é permitido a um recurso realizar um número limitado de tarefas. Por exemplo, num banco não é permitido ao funcionário da caixa efectuar uma hipoteca. Normalmente, uma tarefa apenas pode ser realizada por um número limitado de recursos. Devido ao facto de ser impraticável a indicação de quais os recursos que são capazes de efectuar cada uma das tarefas, classificamo-los usando *classes de recursos*. Uma classe de recursos é um grupo de recursos. Por

exemplo, a classe de recursos, *Empregados_Balcão* pode consistir na Ana, no Horácio, na Maria, no João e no Tomás. Um recurso pode pertencer a mais do que uma classe. Poderíamos supor que a Ana é membro tanto da classe *Empregados_Balcão* bem como da classe de *Agente_Viagens*. Em geral, diferenciamos entre duas formas de classificação de recursos: (1) aquelas baseadas em propriedades funcionais e (2) aquelas baseadas na posição dentro da organização.

Uma classe de recursos baseada na funcionalidade é conhecida por *papel*. É também referida como uma função ou qualificação. Um papel é um grupo de recursos onde cada um deles tem um número de competências específicas. As classes de recursos tais como *Empregados_Balcão, Agente_Viagens, Avaliador, Executivo_Empresa, Administrador, Impressora, Cama_Hospital* e *Doutor_Estagiário* são exemplos de papéis. Ao associar uma tarefa ao papel correcto, podemos garantir que o recurso que irá realizar a tarefa é suficientemente qualificado (e autorizado).

Os recursos também podem ser classificados de acordo com a sua posição na organização. Caem dentro desta definição classes de recursos tais como *Departamento_Vendas, Departamento_Compras, Equipa_2* e *Sucursal_Atlanta*. Uma classe de recursos baseada em características organizacionais, ao invés de características funcionais, é também conhecida como *unidade organizacional*. Esta forma de classificação pode ser usada para garantir que uma tarefa é executada no local certo da organização.

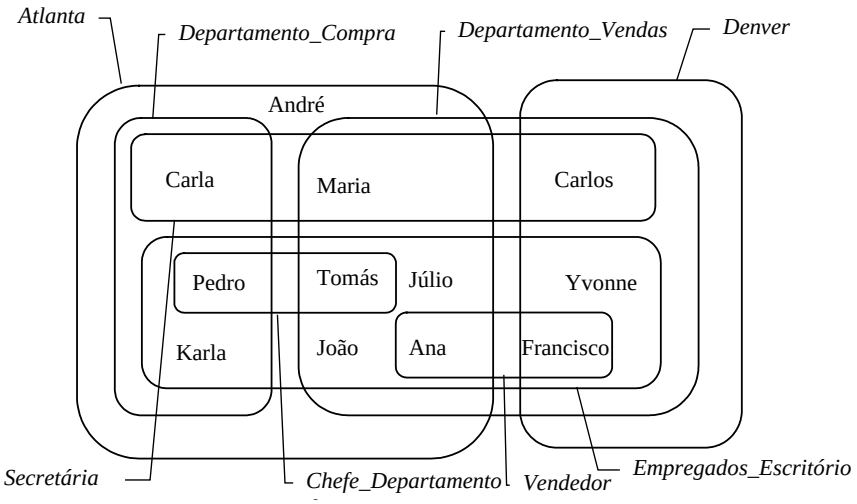


Figura 3.1. Classificação de recursos

A figura 3.1 mostra uma classificação de recursos representada de uma forma gráfica. No total existem oito classes de recursos. De entre estas, as classes de recursos *Atlanta, Denver, Departamento_Compras* e *Departamento_Vendas* são exemplos de unidades organizacionais. Portando o recurso João trabalha no *Departamento_Vendas* na sucursal de *Atlanta*. As restantes classes de recursos são baseadas em características funcionais. A classe de recursos *Secretária*, por exemplo, contém todos os recursos que estão qualificados para agirem como

uma secretária. Como podemos ver na figura 3.1, as classes de recursos podem sobrepor-se. É ainda possível que uma classe de recursos seja um subconjunto de uma classe maior. A classe de recursos *Vendedor*, por exemplo, está contida inteiramente dentro da classe de recursos *Empregados_Escritório*. Podemos usar uma classificação similar à mostrada na figura 3.1 para associar uma tarefa particular aos recursos apropriados. Imaginemos que precisamos de um vendedor em Denver. Neste caso, apenas um recurso está qualificado: Francisco. Se necessitarmos de uma secretária no *Departamento_Vendas,* dois recursos são possíveis*:* Maria e Carlos.

Como já indicámos, na maior parte dos casos uma classificação de recursos consiste em duas partes. Chamamos à parte que contem a estrutura funcional de *modelo do papel* e à parte que contem as unidades organizacionais de *mapa da organização*. Note que o termo mapa da organização normalmente tem um sentido mais amplo, referindo-se à estrutura hierárquica da organização.

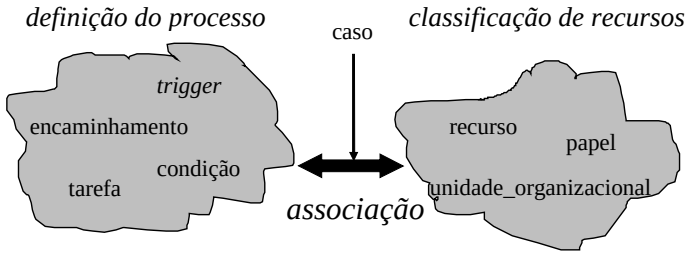


Figura 3.2. Os princípios de associação relacionam a definição do processo com a classificação de recursos

3.1.3 Associando actividades aos recursos

De modo a garantir que cada actividade seja realizada por um recurso apropriado, fornecemos a cada tarefa na definição do processo um *princípio de associação* (ver figura 3.2). Isto especifica as pré-condições que o recurso tem de ter. Na maior parte dos casos, a associação especifica tanto o papel como a unidade organizacional. Nessa altura, o recurso tem de pertencer à intersecção dessas duas classes de recursos. Contudo também é possível definir uma associação muito mais complexa. Na figura 3.1, por exemplo, poderíamos especificar a classe de recursos *Empregados_Escritório* e *Atlanta* mas excluir *Vendedor*. Uma tarefa com esta regra de associação só pode ser executada por um empregado de escritório em Atlanta e que não seja um vendedor. Uma associação também pode depender dos atributos do caso para o qual a tarefa tem de ser executada. Dependendo destes atributos podemos, por exemplo, seleccionar a unidade organizacional. Para avaliar uma declaração de seguro, por exemplo, escolheríamos a sucursal da companhia mais perto. Num caso destes, usaríamos o endereço do cliente como um atributo do caso. Quando a Repartição de Finanças trata de uma declaração de impostos, a associação pode depender do nome da pessoa que está a efectuar a declaração. Uma equipa de

avaliação específica é seleccionada com base no nome. Neste caso, é claro que é o nome da pessoa que age como um atributo do caso.

Ao usar cuidadosamente os atributos de um caso, podemos também garantir que uma actividade é realizada por um recurso específico. Mas, o inverso também é possível. Num banco, por exemplo, pode acontecer que um dos funcionários não tenha permissões para realizar duas tarefas consecutivas num determinado caso. Chamamos a isso *separação de funções*. Este termo é retirado da área da contabilidade. Aqui, é importante que certas tarefas não sejam executadas pela mesma pessoa de modo a prevenir fraudes. O reembolso de um pedido de despesas de viagem, por exemplo, não deve ser efectuado pela pessoa que autorizou a viagem. O objectivo da separação de funções é a de combater os abusos. Como cada caso é tratado por diversas pessoas, torna-se mais difícil de cometer fraudes. Se um número sucessivo de tarefas precisam de ser executadas por, ou sob a autoridade de um único funcionário, então essa pessoa é denominada de *gestor de um caso*. Como um gestor é largamente responsável por um caso, está naturalmente mais envolvido. A atribuição de um gestor para cada caso pode resultar num melhor serviço para o cliente e numa conclusão mais rápida devido a uma maior familiaridade com o trabalho.

Ao dotarmos uma tarefa com um princípio de associação, especificamos as pré-condições que o recurso tem de atingir. Na maior parte dos casos, existe mais do que um recurso que pode levar a cabo a actividade associada a um item de trabalho específico.

O coração de qualquer sistema de *workflow* é o *motor de workflow*. Este garante a execução real de um determinado *workflow*. Uma das suas responsabilidades essenciais é a associação itens de trabalho a recursos. Ao efectuar uma associação, precisa de ter em conta as classes de recursos especificadas, bem como outros elementos tais como a separação de funções e a gestão de casos. Em muitos casos, o motor de *workflow* é capaz de escolher entre um número de recursos aquando da associação de trabalho. Após essa escolha, precisa de decidir qual o recurso que irá executar a actividade. Iremos voltar a este ponto mais à frente.

3.2 Gestão de Recursos em Mais Detalhe

A associação de recursos às actividades não é um processo simples. Como já vimos, conceitos como tarefa, caso, item de trabalho, actividade, atributos de caso, recurso, classe de recurso, papel, unidade organizacional e associação estão intimamente interligados. De modo a explicar estes conceitos de uma forma mais simples, usamos um modelo de dados simples que resume os conceitos e as suas inter-relações. A figura 3.3 mostra um Diagrama Entidade-Relação (DER). Falando de um modo geral, este diagrama consiste em dois tipos de elementos: entidades e relações. O primeiro é ilustrado por meio do um rectângulo e representa um grupo de entidades. Por exemplo, a entidade *tarefa* contem todas as tarefas que compõem um processo. As relações são ilustradas através de um losango. Isto representa um grupo de relações. Assim sendo, a relação *pertence_a*, por exemplo, contem uma colecção de relações entre recursos e classes de recursos. Se existir uma relação entre o recurso *r* e uma classe de recurso *c*, isto significa que *r* pertence a *c*.

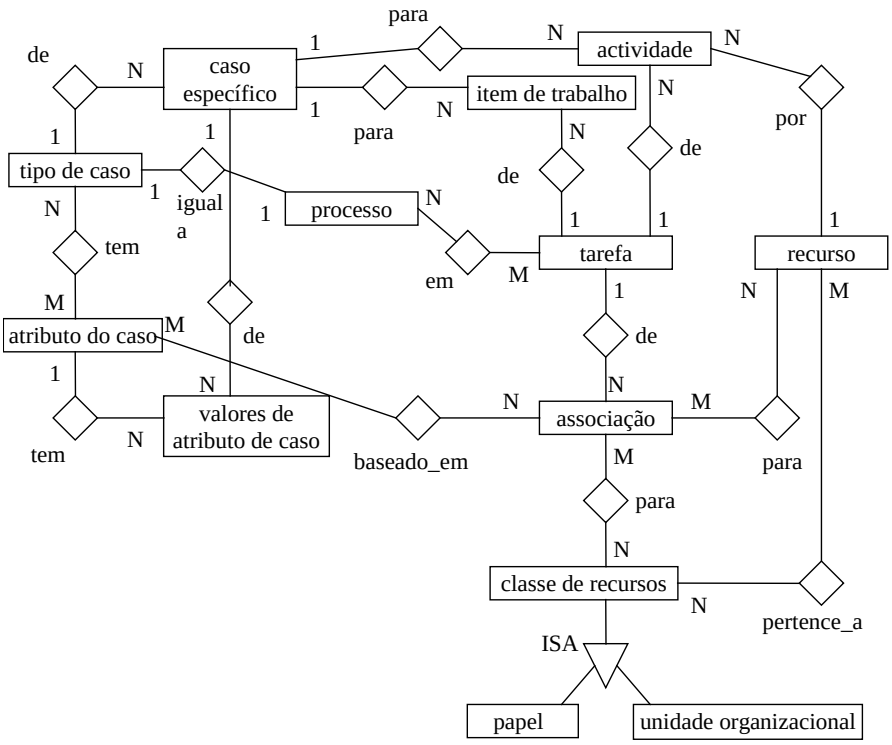


Figura 3.3. Usando um DER, podemos ilustrar as ligações entre as várias entidades

Na figura 3.3, a relação *de* entre *tarefa* e *item de trabalho* indica a que tarefa um item de trabalho pertence. Cada item de trabalho tem uma relação com, exactamente, uma tarefa e cada tarefa pode ter um número arbitrário de itens de trabalho (digamos N) associados a ela. Isto é representado usando os símbolos 1 e N. Assim sendo, estes referem-se à cardinalidade da relação *de*. Podemos também dizer que existe uma relação de 1 para N. Por outras palavras, cada item de trabalho relaciona-se com precisamente um só caso. Pode ser possível que mais do que um item de trabalho tenha uma relação com o mesmo caso. Isto pode, por exemplo, estar relacionado com o encaminhamento paralelo.

Uma entidade do tipo *actividade* está relacionada com o desempenho actual de um item de trabalho. Tal como um item de trabalho, uma actividade está relacionada com um único caso e uma única tarefa. Além disso, zero ou um recurso pode estar também associado a cada actividade. A relação *pertence_a* é um exemplo de um relacionamento de M para N, a qual indica que um recurso pode pertencer a várias classes e uma classe pode conter vários recursos. Um papel e uma unidade organizacional são exemplos de classes de recursos. Assim, os tipos de entidade *papel* e *unidade organizacional* estão associados com a entidade *classe de recursos* por meio de um tipo de relação chamado ISA. Isto indica que papéis e unidades organizacionais são casos especiais de classes de recursos.

No DER, diferenciamos entre um caso específico e um tipo de caso. Este último corresponde a um processo: é a categoria dos casos que podem ser

tratados por esse processo. O DER também indica a existência de uma relação de um para um entre o tipo de caso e o processo. Diferenciamos também entre atributos de um caso e atributos de um caso específicos associados a um caso específico. O primeiro refere-se a um nome lógico que expressa uma propriedade específica, enquanto que o último trata do valor de um atributo num caso específico que se encontra em execução. A entidade *associação* determina as condições que devem ser satisfeitas pela relação *por* entre as entidades *actividade* e *recurso*.

Como foi referido anteriormente, as pré-condições formuladas na política de associação podem tornar-se bastante complexas. Afinal, uma associação interrelaciona tarefas, classes de recursos, atributos de caso e recursos. Cada tarefa tem uma ou mais associações e uma associação pode depender de um ou mais atributos. Na maior parte dos casos, uma associação indicará a intersecção de um papel e de uma unidade organizacional. Nalguns casos especiais, no entanto, um recurso específico pode ser excluído (separação de funções) ou escolhido (gestor de caso).

O DER apenas pode fornecer uma visão dos aspectos *estáticos* da gestão de recursos. Podemos considerar esse diagrama como uma "fotografia" da gestão dos recursos num momento específico, isto é, o diagrama apenas descreve a estrutura de todos os estados possíveis. Os aspectos *dinâmicos* não podem ser ilustrados na figura 3.3. Para os mostrar, temos de olhar para o processo indicado na figura 3.4.

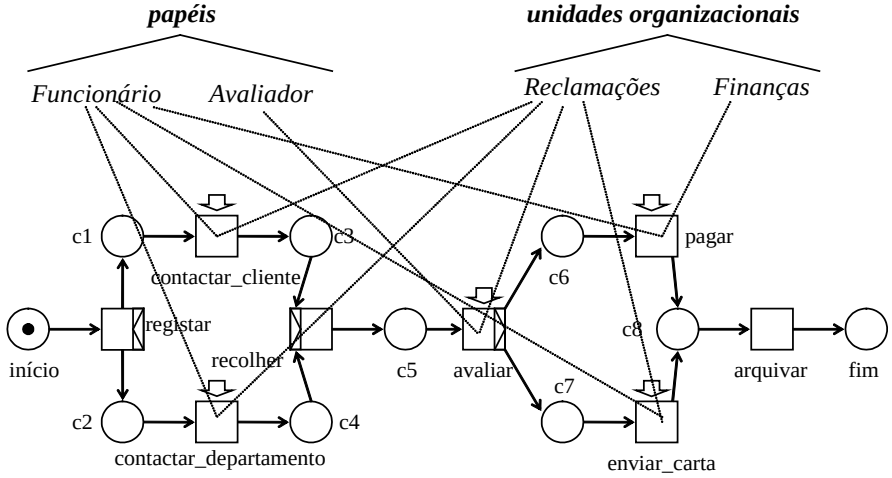


Figura 3.4. O processo "tratar reclamação" e as classes de recurso envolvidas

O processo *tratar reclamação* é composto por oito tarefas, das quais três são executadas automaticamente (não requerem intervenção de nenhum recurso). Além disso, existem quatro classes de recursos. Duas destas são baseadas em características funcionais: *Funcionário* e *Avaliador*. Ao lado destes dois papéis existem ainda mais duas classes de recursos baseadas em características organizacionais: *Reclamações* e *Finanças*. Estas correspondem a dois dos departamentos da organização. A figura 3.4 mostra também, através de um

diagrama, a associação para cada tarefa. A tarefa *contactar_cliente* está ligada ao papel *Funcionário* e à unidade organizacional *Reclamações*. Isto significa que é necessário um funcionário no departamento das reclamações para falar com o cliente. Um recurso resultante da intersecção da classe de recursos *Funcionário* e *Reclamações* é também necessário para as tarefas *contactar_departamento* e *enviar_carta*. Para a tarefa *pagar*, é necessário um funcionário do departamento financeiro. A tarefa *avaliar* é executada por um recurso da intersecção da classe de recursos *Avaliador* e *Reclamações*. Na figura 3.5, estas associações são indicadas outra vez, mas sobre a forma de uma tabela. A composição de cada classe de recursos também é dada.

Na figura 3.5, podemos ver, por exemplo, que a Maria pertence à classe de recursos *Funcionário, Avaliador* e *Reclamações*. Assim sendo, ela pode realizar qualquer tarefa excepto *pagar*. Por outro lado, a Luísa e o Joaquim apenas podem efectuar a tarefa *pagar*.

Classe de recursos	Recursos
Funcionário	João
	Júlio
	Luísa
	Joaquim
	Maria
	Carlos
Avaliador	Maria
	Carlos
Reclamações	João
	Joaquim
	Maria
	Carlos
Finanças	Luísa
	Joaquim

Tarefa	Papel	Unidade organizacional
registar	-	-
contactar_cliente	Funcionário	Reclamações
contactar_departamento	Funcionário	Reclamações
recolher	-	-
avaliar	Avaliador	Reclamações
pagar	Funcionário	Finanças
enviar_carta	Funcionário	Reclamações
arquivar	-	-

Figura 3.5. Um resumo da composição de cada classe de recursos e dos recursos necessários para cada caso

A figura 3.6 mostra os estados de seis casos. O caso 1 foi avaliado positivamente, resultando num item de trabalho (*pagar*). (Por outras palavras, a tarefa *pagar* está pronta para o caso 1). No caso 2, a tarefa *avaliar* está a ser realizada. Baseando-nos nos estados da figura 3.6, podemos estabelecer os itens de trabalho e actividades relevantes. Estes são indicados na tabela da figura 3.7. Contudo, o oposto não é possível. Tendo por base a tabela da figura 3.7 não podemos discernir correctamente o estado de cada caso. Por exemplo, é impossível inferir directamente da tabela que existe um testemunho no local correspondente à condição *c3*.

itens de trabalho	
caso	tarefa
caso 1	pagar
caso 3	avaliar
caso 5	contactar_cliente
caso 5	contactar_deptartamento

actividades		
caso	tarefa	recurso
caso 2	avaliar	Maria
caso 4	contactar_departamento	João
caso 6	registo	-

Figura 3.7. Os itens de trabalho e as actividades para o estado da figura 3.6

Existe um total de quatro itens de trabalho. Cada um corresponde ao desempenho potencial de uma tarefa para um caso particular. Note-se que na situação descrita na figura 3.6 existem dois itens de trabalho para o caso 5. Isto é devido ao encaminhamento paralelo, o qual permite colocar em simultâneo as tarefas *contactar_cliente* e *contactar_departamento* no estado pronto. Existem três actividades. Cada uma delas corresponde ao desempenho actual da tarefa para um caso específico. A primeira corresponde ao desempenho da tarefa *avaliar* para o caso 2 pelo recurso Maria. A segunda é executada pelo João: a tarefa *contactar_departamento* para o caso 4. A última é a tarefa *registar* para o caso 6. Como ilustrado pela figura 3.5, não é necessário nenhum recurso para esta tarefa.

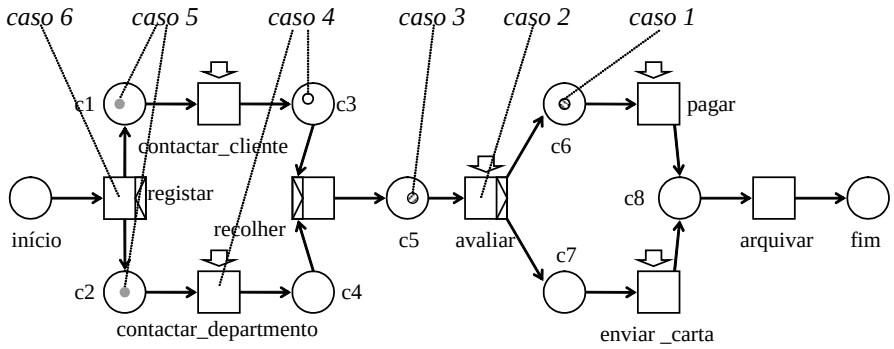


Figura 3.6. No estado ilustrado, existem seis reclamações em progresso

Cada um dos itens de trabalho ilustrados na figura 3.7 pode, em princípio, ser transformado numa actividade. O primeiro (a tarefa *pagar* para o caso 1) requer um recurso que resulta da intersecção da classe de recursos *Funcionário* e *Finanças*. Tanto a Luísa como o Joaquim pertencem a essas classes de recursos. O segundo (a tarefa *avaliar* para o caso 3) apenas pode ser executado por um recurso que resulta da intersecção de *Avaliador* e *Reclamações*. Uma vez que a Maria já está ocupada avaliando o caso 2, o Carlos é o único recurso capaz de executar este item de trabalho imediatamente. Os outros dois itens de trabalho requerem um recurso resultante da intersecção de *Funcionário* e *Reclamações*.

3.2.1 Princípios de associação

O objectivo de um sistema de *workflow* é o de completar itens de trabalho o mais rapidamente possível. Afinal de contas, uma demora que afecte os itens de trabalho pode provocar um atraso do caso no seu todo. De modo a transformar os itens de trabalho em actividades, é sempre necessário tomar duas decisões:

- *Em que ordem é que os itens de trabalho são transformados em actividades?* Se existe um excesso de itens de trabalho em alturas específicas, não podemos imediatamente transforma-los em actividades. Isto deve-se ao facto de poderem existir mais itens de trabalho do que recursos disponíveis. Se esse for o caso, então temos de tomar uma decisão em relação à ordem pela qual os itens são seleccionados.
- *Quais são os recursos que devem executar as actividades?* Uma vez que nem todos os recursos são os mesmos, pode ser importante a associação entre um recurso e um determinado item de trabalho. Um recurso especialista, por exemplo, pode desempenhar certas tarefas mais rapidamente. Poderá ser também sensato manter um recurso flexível (um recurso que é membro de um grande número de classes de recursos) livre durante o maior tempo possível.

Escusado será dizer que estas duas decisões têm uma interrelação bastante forte. A ordem pode ser importante aquando da selecção de um recurso. Inversamente, a escolha de um recurso pode afectar a ordem pela qual os itens de trabalho são transformados em actividades.

Podem ser aplicadas muitas heurísticas diferentes para seleccionar uma ordem específica. Em particular, podemos recorrer às várias *políticas de gestão* de filas usadas na gestão de produção em fábricas. O encaminhamento de um caso através de vários recursos demonstra muitas semelhanças com o encaminhamento de um produto através das máquinas num departamento de produção. Seguem-se algumas das políticas de gestão de filas mais comuns:

- *First-In, First-Out (FIFO).* Se os itens de trabalho são tratados pela ordem pela qual são criados, referimo-nos a uma política FIFO. Ao invés de usarmos o tempo no qual o item de trabalho foi criado, podemos também usar o momento em que o caso como um todo foi criado. A política FIFO é uma regra de associação simples e robusta e é a mais usada na prática.
- *Last-In, First-Out (LIFO).* A política LIFO é o oposto da política FIFO. Neste arranjo, os itens de trabalho criados mais recentemente são tratados primeiro. Nalguns casos, esta regra de associação (injusta) pode conduzir a um nível médio mais elevado de serviço.
- *Shortest Processing Time (SPT).* Podemos às vezes estimar à priori, usando os atributos de um caso, quanto tempo será necessário para realizar uma actividade. Muitas vezes pode ser feita uma distinção entre casos fáceis e difíceis, e entre tarefas simples ou muito demoradas. Ao seleccionarmos primeiro os itens de trabalho que necessitam de menos tempo, é muitas vezes possível reduzir o tempo médio do fluxo dos casos. Contudo, também é possível imaginar situações nas quais é melhor dar prioridade às tarefas

mais demoradas sobre as mais simples. Neste caso referimo-nos a uma política de gestão *Longest Processing Time* (LPT).

- *Shortest Rest-Processing Time (SRPT).* Se temos alguma ideia sobre o tempo necessário para realizar actividades específicas para um determinado caso, e sobre o encaminhamento desse caso, podemos então estimar o tempo de processamento restante. Ao darmos sempre prioridade ao caso com o mais curto tempo de processamento restante, a quantidade de *trabalho em progresso* (TEP) é geralmente minimizada. Pelo contrário, se escolhemos o caso com o maior tempo de processamento restante, estamo-nos a referir a uma política de gestão *Longest Rest-Processing Time* (LRPT).
- *Earliest Due Date (EDD).* Uma actividade é sempre executada no contexto de um caso. Uma actividade é iniciada numa determinada altura, e deve ser também completada numa determinada altura (a "data limite"). A política EDD determina a ordem tendo por base a data limite do caso. Portanto, um caso que tem de estar completado hoje tem prioridade sobre um caso que tem de estar pronto dentro de uma semana. As tarefas que ainda precisam de ser executadas podem também ser tomadas tendo em conta aquando da decisão da ordem.

Note-se que a informação necessária por cada política de gestão de filas pode variar bastante. A FIFO não precisa de quase nenhuma informação. A SRPT, por outro lado, precisa de informação relacionada com o tempo de processamento esperado e o encaminhamento. Existem também políticas de gestão de filas bastante avançadas que têm em conta o trabalho em progresso, o trabalho esperado e a disponibilidade dos recursos. Estas políticas são caracterizadas pelo uso do estado corrente do *workflow* ou de previsões do seu estado futuro.

Ao considerarmos políticas de gestão de filas, temos assumido até agora que a ordem é determinada pelas características individuais de um caso. Contudo, também é possível que ela seja escolhida tendo por base grupo de casos. Para um dado grupo, é por vezes possível melhorar a ordem usando certos critérios.

A ordem pela qual os itens de trabalho são transformados em actividades está associada de perto com a escolha dos recursos. Se um item de trabalho pudesse ser executado por mais do que um recurso, então teríamos de ter em conta as seguintes considerações:

- *Deixar um recurso executar a sua especialidade.* Muitas vezes um recurso pode efectuar múltiplas tarefas. Contudo, normalmente existem algumas em que ele se especializa. Por exemplo, um inspector de impostos pode ser qualificado para avaliar uma gama completa de declarações de impostos, mas ao mesmo tempo ser especializado em declarações de impostos de empresas de construção. É portanto preferível deixar este recurso executar a sua especialidade.
- *Sempre que possível deixar um recurso efectuar tarefas semelhantes sucessivamente.* Tanto as pessoas como as máquinas precisam de um "tempo de iniciação" ou seja, o tempo (adicional) necessário para começar a efectuar uma nova tarefa. O tempo de iniciação pode, por exemplo, ser gasto para abrir uma aplicação de *software* ou então para uma pessoa se ambientar a uma nova tarefa. Ao efectuar tarefas semelhantes uma após a outra, este tempo pode ser reduzido. Além disso, no caso de um trabalho de natureza

repetitiva, as pessoas podem reduzir o seu tempo médio de processamento ao entrar na rotina.

- *Tentar alcançar a maior flexibilidade possível para o futuro próximo.* Se temos uma escolha entre dois recursos de igual valor para realizar um item de trabalho, é sensato seleccionar aquele que pode efectuar menos itens de trabalho de outros tipos. Por outras palavras, devemos guardar os "generalistas" para o fim. Na situação indicada na figura 3.7, por exemplo, não seria sensato associar o Carlos a um dos itens de trabalho do caso 5. Se o fizéssemos, todos os recursos da classe de recursos *Avaliador* ficariam ocupados e o caso 3 não poderia avançar mais. Ao manter os "generalistas" livres, garantimos um nível desejável de flexibilidade para o futuro próximo.

Portanto, ao associarmos itens de trabalho a recursos, temos de efectuar escolhas continuamente. Existem duas maneiras de realizar isso:

- *O motor de workflow associa itens de trabalho a recursos.* Dentro das pré-condições, o motor de *workflow* pode escolher qual o recurso que realiza cada item de trabalho. Assim, o próprio recurso é incapaz de escolher. Logo que ele acaba de efectuar uma actividade, é-lhe dado um novo item de trabalho. Este comportamento é referido de *push-driven*: o motor "empurra" itens de trabalho aos recursos
- *Os recursos seleccionam os itens de trabalho.* Neste cenário são os próprios recursos que tomam a iniciativa. Cada um deles, conhece os itens de trabalho que consegue realizar, e escolhe-os. Chamamos a este comportamento *pull-driven*: os recursos "puxam" os itens de trabalho e todos eles "consomem" itens de trabalho do mesmo conjunto.

Normalmente é adoptada uma aproximação entre o *push* e *pull-driven*. Um método comum é usar o princípio de *pull* complementado por uma ordenação dos itens de trabalho por parte do motor de *workflow*. Assim, um recurso vê uma lista ordenada de itens de trabalho que pode realizar. Esta lista é fornecida pelo motor de *workflow*, o qual ordena os itens de trabalho de acordo com princípios tais como o FIFO, o LIFO, o SPT ou o EDD. Os recursos seleccionam preferencialmente o primeiro item de trabalho da lista. Contudo, eles podem (por qualquer razão) escolher outro item. A vantagem desta aproximação mista é que o motor de *workflow* fornece um aconselhamento enquanto que os recursos (humanos) retêm ainda a liberdade para decidir qual o trabalho que irão realizar.

3.3 Melhorar Workflows

Um sistema de *workflow* possibilita a uma organização usar e gerir processos de negócio estruturados. Uma propriedade importante dos sistemas de *workflow*, em comparação com os sistemas de informação clássicos, é que se torna mais fácil a organização mudar os seus processos de negócio. Trocar e combinar tarefas, ou reajustar classes de recursos são modificações fáceis. Torna-se, por isso, interessante examinar como podemos melhorar os *workflows* geridos pelo sistema. Os melhoramentos influenciam os critérios de desempenho tal como o

tempo de conclusão, a utilização da capacidade, o nível de serviço e a flexibilidade.

3.3.1 Congestionamentos nos workflows

Quando é que um processo, a classificação de recursos ou a gestão de recursos devem ser alterados? Se um *workflow* não está a funcionar correctamente, conseguimos muitas vezes identificar vários tipos de sintomas. Podemos compará-los com as nossas funções corporais. Sintomas como dores de cabeça, diarreia, náuseas ou tosse indicam problemas. Num *workflow*, também existem sintomas típicos que evidenciam a presença de congestionamentos que impedem o seu funcionamento normal. Alguns sintomas típicos são os seguintes:

- *Número de casos em progresso (demasiado) elevado.* Se existem bastantes casos em progresso, pode ser a indicação de um problema. Este número elevado pode ser causado por grandes flutuações no fornecimento de casos ou por falta de flexibilidade nos recursos. Contudo, também pode ser devido ao processo conter um elevado número passos que necessitam de ser efectuados sequencialmente.
- *Tempo de conclusão (demasiado) longo comparado com o tempo de processamento efectivo.* Às vezes, o tempo efectivo de processamento de um caso é apenas uma pequena parte do tempo total em que este está em execução. Caso isto aconteça, poderá existir uma diversidade de possibilidades para reduzir o tempo de conclusão.
- *Nível de serviço (muito) baixo.* O nível de serviço de um *workflow* é o grau com o qual uma organização é capaz de completar casos dentro de um prazo limite. Se o tempo de conclusão varia num grande intervalo, então existe um baixo nível de serviço. Não é possível garantir um determinado tempo de conclusão. Um baixo nível de serviço também existe quando ocorrem bastantes "no sales". (Com isto, queremos dizer que existe uma incapacidade de iniciar potenciais casos devido a longos tempos de espera). Quando um cliente sabe que irá levar muito tempo para completar um caso (por exemplo, um empréstimo), irá procurar outra empresa. Um baixo nível de serviço pode indicar uma falta de flexibilidade, um processo mal desenhado ou uma falta de capacidade estrutural.

Estes três sintomas indicam possíveis congestionamentos. Para identificá-los necessitamos de valores de referência para estas medidas, por exemplo, de processos comparáveis. Usualmente, não é sensato combater estes sintomas usando apenas medidas de emergência. É importante atacar as suas causas.

Para alertar-nos sobre potenciais problemas e para medir o desempenho de um *workflow* em particular, usamos *indicadores de desempenho*. Estes indicam o desempenho do *workflow* relativamente a um aspecto particular. Em geral, fazemos uma separação dos indicadores de desempenho em dois grupos:

- *Indicadores de desempenho externos (focados nos casos).* Os indicadores de desempenho externos focam aqueles aspectos que são importantes para o ambiente do *workflow*. Por exemplo, indicadores do tempo de conclusão

médio e do nível de confiança do tempo de conclusão. Note que estes indicadores podem ser subdivididos de acordo com as propriedades específicas do caso.

- *Indicadores de desempenho internos (focados nos recursos)*. Os indicadores de desempenho internos mostram que esforços são necessários para alcançar o desempenho externo (por exemplo, o nível de utilização dos recursos, o número de casos por recurso, o número de casos em progresso, o número de *rollbacks*, e a taxa de rotatividade). Este último é uma medida que indica a velocidade dos casos que prosseguem através do sistema de *workflow*. É calculado dividindo a duração de um período (por exemplo, um mês) pelo tempo médio de conclusão, ou dividindo o número médio de casos que ocorrem durante um período pelo número médio de casos em progresso.

Um desempenho externo deficiente implica uma despesa elevada. Vejamos, por exemplo, um banco: um tempo de conclusão longo para um pedido de empréstimo leva à perda de muitos clientes. Todavia, um bom desempenho externo pode exigir um alto grau de esforço interno. Alcançar um rápido tempo de conclusão pode, por exemplo, exigir tempo extra ou a associação de recursos adicionais. O objectivo de cada organização é minimizar o seu custo total. Como está ilustrado na figura 3.8, é necessário uma ponderação cuidada dos custos de um fraco desempenho externo (custo de "no-sale") versus os custos de esforço interno requeridos.

Não obstante é possível em muitos casos melhorar o desempenho externo de um *workflow* sem associar recursos adicionais. Tal melhoramento pode ser alcançado através da reestruturação do *workflow* ou usando uma melhor estratégia de associação.

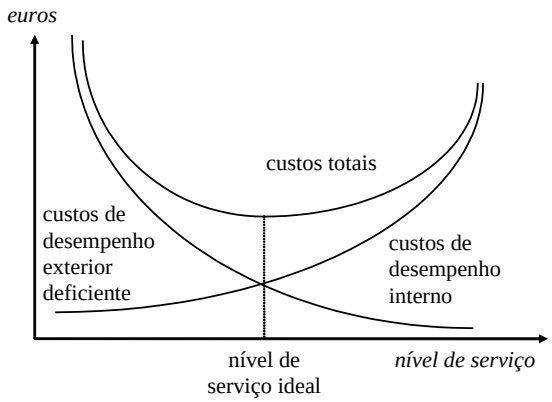


Figura 3.8. Ponderar o desempenho externo versus o esforço interno

3.3.2 Reengenharia de Processos de Negócio

Antes de abordar a forma de melhorar *workflows*, iremos considerar as relações entre reengenharia de processos de negócio (BPR – *Business Process Re-engineering*) e gestão de *workflow*. Podemos definir o BPR como a

reconsideração fundamental dos processos de negócio. O seu objectivo é trazer novos processos de negócio que permitem melhoramentos drásticos nos custos, qualidade e serviço. De forma a atingir este objectivo, são muitas vezes necessárias mudanças radicais. Para muitos processos administrativos, a implementação de sistemas de gestão de *workflow* é um "catalizador essencial" para esforços BPR. Afinal de contas, o uso de um sistema de gestão de *workflow* torna fácil adaptar processos.

A introdução de um sistema de *workflow* torna possível trabalhar de um modo completamente diferente. Reciprocamente, alguns esforços BPR resultam na compra de um sistema de gestão de *workflow*. Gestão de *workflow* e BPR são parceiros naturais. É portanto importante que *designers* de processos de trabalho estejam atentos aos últimos desenvolvimentos em BPR.

No seu livro *Re-engineering the Corporation*, Michael Hammer e James Champy escrevem que o BPR é caracterizado por quatro palavras-chave: fundamental, radical, dramático e processo. A palavra-chave *fundamental* indica que é de grande importância, quando revitalizando um processo de negócio, perguntar sempre as questões elementares: porque estamos a fazer isto, e porque estamos a fazê-lo desta forma? *Radical* significa que a reengenharia tem de representar uma separação total do modo de trabalhar actual. O BPR não é um melhoramento dos processos existentes, mas a sua substituição por outros completamente novos. A terceira palavra-chave também se refere ao facto do BPR não apenas realizar mudanças meramente marginais ou superficiais, mas que deve ser *dramático* em termos de custos, serviço e qualidade. Mas de todas as palavras-chave, *processo* é talvez a mais importante. De modo a atingir um melhoramento drástico, é necessário concentramo-nos nos processos de negócio. Isto significa que a organização tem de estar subordinada aos processos de negócio primários. Para actuar de uma maneira verdadeiramente orientada aos processos, temos de nos abstrair de outros aspectos, tais como as pessoas, as funções, as equipas e os departamentos.

O pensamento orientado ao processo é crucial na utilização de um sistema de gestão de *workflow*. Um dos maiores perigos que ameaça a introdução com sucesso de um sistema de *workflow* ocorre quando se decide efectuar unicamente uma simples computorização/automatização de práticas (manuais) existentes. Suportar processos antigos num sistema de *workflow* resulta em melhoramentos limitados. Melhoramentos drásticos são apenas possíveis se os processos antigos forem separados e substituídos por novos. Um erro comum aquando da introdução de um sistema de *workflow* é a especificação desnecessária de tarefas em sequência. O facto que um documento físico só poder estar num determinado local, num determinado momento, conduze à definição obrigatória de encaminhamentos sequenciais em muitos processos antigos. Contudo, a computorização de documentos e o uso de sistemas de *workflow* tornou possível em muitos casos um encaminhamento paralelo. É importante estruturar o novo processo para que o encaminhamento paralelo também se torne possível (ver o capítulo 6).

3.3.3 Directrizes para (re)desenhar workflows

Inspirados pela nossa experiência em BPR, somos capazes de propor um número de boas práticas para o desenho ou redesenho de *workflows*. Estas

referem-se ao desenho de processos, classificação de recursos e à atribuição de actividades a tarefas:

1. *Primeiro é necessário estabelecer o objectivo do processo*. Quando estamos a desenhar um novo *workflow* ou a mudar um existente, é crucial considerar o papel desempenhado pelo processo num plano mais elevado. Porque precisamos do *workflow*? Reflectindo sobre esta questão fundamental é possível definir um novo *workflow* sem pressuposições falaciosas.

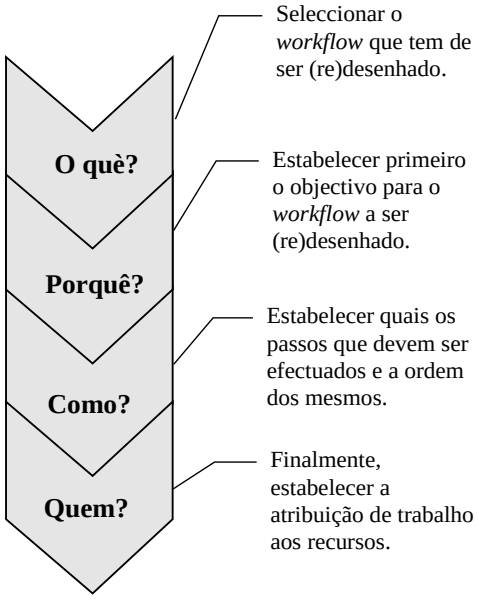


Figura 3.9. As quatro fases que o (re)desenho de um *workflow* atravessa

2. *Ignorar a existência de recursos enquanto estivermos a definir um processo*. A definição de um processo é independente do potencial oferecido pelas pessoas e máquinas. Se já considerarmos a atribuição de trabalho a recursos quando desenhamos a definição do processo, corremos o risco do processo resultante não ser o melhor possível. Primeiro devemos listar quais as tarefas que são necessárias e em que ordem devem ser executadas. Só depois devemos associar as tarefas a recursos. Por outras palavras, não nos devemos deixar distrair pela estrutura tradicional da organização quando definimos um processo. Ao todo, reconhecemos quatro fases no (re)desenho de um *workflow*: (1) *O quê?*, (2) *Porquê?*, (3) *Como?*, e (4) *Quem?*. A figura 3.9 mostra graficamente estas fases. Durante a primeira fase escolhemos o processo que precisa de ser redesenhado. Na segunda, consideramos o objectivo do processo: qual é o seu resultado, em termos do produto entregue, e sobre a sua necessidade? Durante a terceira fase, determinamos a estrutura do processo. Apenas durante a última fase é que nos preocupamos em atribuir trabalho aos recursos.

3. Tanto quanto possível, tornar uma pessoa responsável pelo processamento de um caso (gestor de caso). Processos suportados por um sistema de *workflow* podem ser bastante complicados. Por consequência, para o cliente, é frequentemente muito difícil medir o progresso de um caso particular. É por isso importante designar um gestor para cada caso. Ele ou ela funciona como uma espécie de intermediário entre o processo complexo e o cliente. Ao fazê-lo, é importante que o gestor de caso se comporte perante o cliente como se fosse responsável por todo o processo. Isto fornece ao cliente um único ponto de contacto, e o gestor de caso sente-se mais envolvido no trabalho. Note-se que o gestor de casos é apenas responsável pelo caso em si. Na realidade, outros recursos podem ser usados para concluir as actividades associadas ao caso.

4. *Verificar a necessidade de cada tarefa.* Às vezes, as tarefas são acrescentadas por motivos de segurança: por exemplo, para monitorizar tarefas. Tais tarefas são frequentemente usadas como solução provisória para esconder um problema numa das tarefas anteriores. Pela mesma razão, as iterações também devem ser examinadas de uma maneira exigente. Por outras palavras, é necessário eliminar as tarefas que não têm utilidade.

5. *Considerar o âmbito das tarefas.* Uma tarefa é uma unidade lógica de trabalho. Ao combinar tarefas separadas numa tarefa composta, o tempo de espera pode ser reduzido. O envolvimento das pessoas que as executam também é reforçado. Contudo, as tarefas não deverão ser muito grandes. São aconselháveis tarefas de uma dimensão apropriada, porque uma tarefa tem de ser realizável de uma só vez, sem interrupções. Tarefas de grandes dimensões também podem inibir a flexibilidade e tornar impossível uma atribuição avançada de trabalho pelos recursos.

6. *Procurar arduamente o processo mais simples possível.* Definições de processos complexos originam processos intratáveis. É por esta razão que é importante que um processo não seja desnecessariamente complicado. Os processos podem muitas vezes ser simplificados adicionando mais "inteligência" às tarefas. Se for impossível evitar um processo complexo, então é essencial estabelecer uma estrutura hierárquica nítida. É importante garantir que tarefas com uma relação próxima formem parte do mesmo sub-processo quando estamos decompondo um processo. Para além disso, é sensato permitir tão poucas ligações causais quanto possível entre diferentes sub-processos. Idealmente, cada sub-processo terá uma entrada e uma saída. Todavia, a consideração mais importante é que o processo seja percebido pelas pessoas que executam o trabalho. Se isto não acontece, o resultado pode ser um processo difícil de gerir.

7. *Ponderar cuidadosamente um processo genérico versus várias versões do mesmo processo.* Não definir um processo separado para cada tipo de caso. Tentar criar um processo genérico que diferencie os vários tipos de casos usando o encaminhamento selectivo. Não tentar, todavia, lidar com dois tipos de casos completamente diferentes num único processo. Se um processo começa com uma divisão OR-split que envia o caso para um conjunto de sub-processos alternativos, então é provavelmente boa ideia usar um conjunto de sub-

processos distintos. Cada um destes irá corresponder a uma versão do mesmo processo.

8. *Ponderar cuidadosamente a especialização versus a generalização*. A divisão de uma tarefa genérica em duas ou mais tarefas alternativas pode ter um efeito positivo ou negativo. Uma vantagem poderá ser o facto de as tarefas ficarem mais ajustadas às características específicas de um recurso. Contudo, podem existir inconvenientes à especialização. Frequentemente, esta diminui a flexibilidade e acessibilidade do processo. Pode também levar à criação de trabalho monótono, o que reduz a motivação. Em vez de usar o termo especialização, o termo "*triagem*" é frequentemente utilizado, isto é, a classificação de casos de maneira a possibilitar um processamento selectivo.

9. *Tanto quanto possível, tentar realizar um processamento paralelo das tarefas*. É necessário ter sempre em consideração se as tarefas podem ser realizadas em paralelo. Se duas tarefas podem ser realizadas independentemente uma da outra, então é muito importante que o processo permita a sua execução paralela. A introdução desnecessária de relações de ordem sequenciais resulta em tempos de conclusão mais longos e ao uso ineficiente de recursos.

10. *Investigar as novas oportunidades disponibilizadas pelos recentes avanços em redes e bases de dados (distribuídas)*. A eliminação de barreiras físicas resultantes de desenvolvimentos, tais como a disponibilização de documentos digitais, possibilita frequentemente a criação de processos com estruturas totalmente novas. Tarefas que tinham de ser executadas em sequência podem agora ser executadas em paralelo pela introdução de, digamos, um sistema de *workflow*.

11. *Tratar recursos geograficamente dispersos como se estivessem centralizados*. A introdução de um sistema de *workflow* reduz as barreiras físicas entre os vários departamentos de uma organização. Torna mais fácil a troca de trabalho entre duas unidades organizacionais. Se a equipa A está comprometida com um excesso de trabalho, mas a equipa B está a funcionar abaixo da sua capacidade, então é lógico transferir trabalho de A para B. É ainda melhor tratar recursos geograficamente dispersos como se estivessem centralizados. Isto torna os recursos capazes de serem atribuídos aos locais onde grande parte do trabalho necessita de ser realizado.

12. *Permitir a um recurso exercer a sua especialidade*. Como mencionado anteriormente, é importante fazer uso das qualidades específicas de um recurso.

13. *Tanto quanto possível, permitir a um recurso realizar sucessivamente tarefas semelhantes*. Ao realizar tarefas semelhantes sucessivamente, os tempos de espera podem ser reduzidos e os benefícios do trabalho rotineiro podem ser explorados.

14. *Tentar alcançar tanta flexibilidade quanto possível para o futuro próximo*. Ao atribuir trabalho aos recursos, é sensato manter tanta flexibilidade quanto possível para um futuro próximo.

15. *Permitir a um recurso trabalhar tanto quanto possível no mesmo caso*. Se um funcionário realiza um número de tarefas sucessivas para um caso específico, o tempo de processamento total é geralmente menor do que se fossem funcionários diferentes a executar essas tarefas. É necessário menos tempo porque o funcionário não tem de se habituar a cada novo caso.

Com base nestas boas práticas, podem ser desenhados *workflows* de forma a resultar no processamento eficiente e efectivo de casos. Uma variedade destas boas práticas destaca o facto de ser necessário um balanço entre duas ou mais alternativas. Em muitos casos, as práticas que devem ser escolhidas devem ser precedidas de uma análise profunda. Tal análise é usualmente realizada sob aspectos quantitativos, com ênfase em indicadores de desempenho como a média de tempos de conclusão, o nível de serviço e a capacidade de utilização. Existem várias técnicas analíticas disponíveis para estabelecer estes indicadores de desempenho usando um *workflow*. Algumas destas são discutidas no próximo capítulo.

EXERCÍCIOS

Exercício 3.1 Companhia de seguros

Considere a companhia de seguros descrita no exercício 2.7.

(a) Faça uma classificação dos recursos estabelecendo relações entre papéis (competências) e grupos (unidades organizacionais).
(b) Atribua um papel e um grupo para cada tarefa do processo.

Exercício 3.2 Manipulação de reclamações

Considere o processo de manipulação de reclamações descrito no exercício 2.8.

(a) Faça uma classificação dos recursos estabelecendo relações entre papéis (competências) e grupos (unidades organizacionais).
(b) Atribua um papel e um grupo para cada tarefa do processo.

Exercício 3.3 Agência de emprego

A agência "*Job Shop*" aceita pedidos de empresas de todo o mundo à procura de novos funcionários. Os pedidos podem ser enviados por *e-mail*, por correio, ou efectuadas por telefone para uma das agências em Eindhoven e Leeuwarden. O tratamento destes pedidos é da responsabilidade de alguém do departamento de Relações Empresariais (RE). Na agência de Eindhoven o responsável é o Johan, em Leeuwarden é o Sietse, que é o responsável pelo RE. A primeira

tarefa a ser realizada é a devolução de uma confirmação a indicar que um pedido foi recebido. Em seguida, o "*Job Shop*" tem várias opções: eles procuram sempre por pessoas que preencham os requisitos na base de dados, mas podem também colocar um anúncio nalguns dos jornais de maior tiragem do país para recrutar pessoas. Colocar um anúncio é uma tarefa para o pessoal do departamento de Relações Públicas (RP). Jaap e Anke em Eindhoven, e Rinske em Leeuwarden. O gerente decide se esta opção vai ou não ser utilizada. O cargo de gerente é desempenhado por Ahmed (Eindhoven) e por Dion (Leeuwarden).

A pesquisa efectiva realizada na base de dados é feita por alguém do recrutamento. Todos os candidatos para o trabalho recebem um identificador que será usado para uma consulta futura.

As pessoas que respondem ao anúncio podem fazê-lo através do telefone, completando um formulário (disponível na Internet), ou deixando uma carta com os seus dados no escritório. Alguém do recrutamento trata dos dados do formulário/carta, adicionando-os à base de dados e atribuindo um identificador aos candidatos. Se o contacto for feito por telefone, um dos membros do recrutamento irá entrevistar essa pessoa de forma a obter os seus dados para a base de dados. De novo, um identificador é colocado se a pessoa preencher os requisitos para o trabalho.

A equipa de recrutamento de Eindhoven é constituída por Annelies, Manja, e pelo pessoal do departamento tanto de RP e RE. Em Leeuwarden, os encarregados do novo pessoal são Anja, Hakan, Rinske (também RP) e Sietse (RE).

Após algum tempo, o prazo para um trabalho expira e um candidato tem de ser escolhido de entre os identificados na base de dados. Posteriores respostas ao anúncio não serão tidas em conta. Um a um, os candidatos serão chamados por alguém da equipa de recrutamento até alguém ser escolhido. A pessoa escolhida receberá um convite por telefone para ir ao escritório discutir a possibilidade de um novo trabalho. É claro que as pessoas podem recusar-se a ir. No entanto, se alguém concordar em ir ao escritório, uma marcação é feita e o potencial candidato realiza uma entrevista com um dos funcionários (do departamento de recrutamento) do "*Job Shop*". Imediatamente após esta entrevista é feita uma avaliação e é comunicado ao candidato se irá ser seleccionado ou não. Se não for encontrado nenhum candidato, ou quando nenhum é apropriado para o trabalho, uma carta é enviada para a empresa.

Assim que alguém for escolhido, essa pessoa recebe uma carta com todas as informações necessárias para se preparar para o novo trabalho. Esta carta é feita por alguém do recrutamento. O departamento de RE envia também uma carta à empresa que vai contratar o novo funcionário. Nesta carta está presente toda a informação relevante acerca do novo funcionário. É claro que a base de dados terá de ser actualizada de forma a reflectir o novo estado do candidato. Isto é feito após o envio das cartas, pela mesma pessoa do recrutamento que as enviou.

A manutenção das bases de dados em ambas as agências é feita pelo especialista em IT, Mahroud.

(a) Faça uma classificação dos recursos estabelecendo relações entre papéis (competências) e grupos (unidades organizacionais).
(b) Construa um modelo para o processo esboçado anteriormente.

Exercício 3.4 Tenha um bom voo com CRASH

Iremos descrever a preparação de um plano de voo para um avião da companhia "CRASH" (*Cheap and Reliable Aerial Shipments*). Esta companhia transporta mercadorias da zona Y para a zona Z.

Cada cliente envia um formulário descrevendo a mercadoria e os seus requisitos quanto à mesma. Aquando da recepção do formulário, a secretária faz uma cópia. O original é levado ao *loadmaster*, que, com o seu conhecimento da capacidade dos aviões, irá decidir qual o avião a ser utilizado. A cópia é enviada ao navegador. O navegador, responsável por delinear o plano de voo, utiliza um formulário de plano de voo e preenche a data, os seus dados (nome e número de funcionário) e o número do cliente. Depois, o navegador deve verificar os seguintes pontos em sequência antes de planear o voo:

- Que mercadorias serão transportadas e, mais importante, onde têm de ser entregues? Isto irá ser discutido em conjunto com o *loadmaster*. O tipo de avião e a sua carga irão influenciar o plano de voo: talvez alguma paragem extra seja necessária para reabastecer.
- Quais são as condições meteorológicas? Para isto, o navegador vai à parte norte do edifício da empresa para reunir-se com alguém do departamento meteorológico. Juntos irão discutir as condições meteorológicas para esse dia e essa pessoa irá colocar a informação num mapa.
- Poderão existir algumas excepções: algumas áreas têm de ser evitadas devido a exercícios militares, etc. No lado sul do edifício encontra-se a sala dos directores. Eles têm conhecimento destas excepções e irão comunicar ao navegador qualquer excepção existente. O mesmo mapa é usado para desenhar áreas onde existam excepções.

Assim que o navegador tiver recolhido estes três elementos de informação, ele pode iniciar o planeamento do voo na sua sala no lado oeste da empresa. Para isto, ele utiliza um formulário especial, e não o formulário que já tinha sido preenchido anteriormente. Ele usa um formulário adicional para poder efectuar alterações sem danificar o plano de voo oficial. Após a conclusão, o plano de voo é levado aos directores. Um deles irá comparar este novo plano com outros planos de voo já aprovados para prevenir colisões com outros aviões devido a conflitos. Possíveis erros feitos pelo navegador, independentemente da sua gravidade, irão ser identificados.

Se o plano de voo é considerado inseguro, o navegador volta à sua sala de modo a iniciar novamente o planeamento e obter um plano de voo melhorado. Isto será procedido de uma verificação com os directores, o número de vezes necessárias até tornar o plano de voo seguro. Só então, o navegador e o director irão assinar o plano de voo, após ter sido preparado um formulário oficial por uma secretária especialmente treinada para o mesmo.

Visto que o combustível tem de ser pago pela própria empresa, um mensageiro tem de levar o plano de voo a um dos funcionários da logística (num outro edifício a dois quilómetros da sala do navegador). Esta pessoa tem de assinar o plano de voo para aprovar o uso do combustível. Claro, ele pode negar-se a assinar. Nesse caso, a recusa irá ser tornada clara ao navegador e será enviada uma carta ao cliente. Nesta carta, a empresa irá expressar as suas desculpas e explicará o porquê de nenhum plano de voo aceitável ter sido

produzido. Obviamente, a "CRASH" espera fornecer um melhor serviço no futuro.

Contudo, se o funcionário da logística aprova, o mensageiro leva de volta o plano de voo. Então, o capitão do avião tem de o assinar. Isto é devido à responsabilidade que ele tem durante cada segundo do voo. Novamente, o plano de voo pode ser recusado, com as mesmas consequências que as anteriormente mencionadas. Se o plano de voo for aceite (por assinatura), o plano de voo irá ser armazenado no computador por um dos directores.

Após uma entrega com sucesso (apesar do nome da empresa, grande parte das entregas o são), o cliente irá também receber uma carta, acompanhada de uma factura. Contudo, por vezes o avião despenha-se. Então uma carta de desculpas é enviada ao cliente. Todas as cartas são redigidas e enviadas por uma secretária.

A partir do momento em que o plano de voo tenha sido aceite para ser assinado pela logística e pelo capitão do avião, o navegador fica disponível para planear outro voo.

Acerca da organização: a grande maioria dos navegadores também são capitães. Consequentemente, todos os capitães e navegadores estão unidos na divisão AIR. (Dizem que AIR significa *Aces with Incredible Reputations*; humildade não é o ponto forte deles). Capitães extra que foram contratados da KLM (*Kaptains Looking for Money*, uma agencia que "tem" capitães e pilotos freelancer) também fazem parte do AIR, embora temporariamente. O suporte terrestre dos *loadmasters*, directores e funcionários da meteorologia está sobre a alçada da divisão SUPPORT: *SUPPort Of Reliable Transport*. Os departamentos de logística e de secretaria são parte integrante da CRASH, mas como eles não foram capazes de arranjar um nome interessante, não têm um grupo só deles. Os mensageiros são contratados a uma agência com relações próximas à empresa.

(a) Construa a classificação dos recursos da CRASH, distinguindo papéis e grupos, usando as técnicas descritas neste livro.
(b) Construa um modelo para o processo descrito anteriormente. Defina os papéis, e atribuía disparos e papéis às tarefas sempre que for apropriado.
(c) Analise o processo e investigue possíveis melhoramentos.

(Página deixada propositadamente em branco)

Capítulo 4.

ANALISAR WORKFLOWS

4.1 Técnicas de Análise

A introdução ou modificação de um processo de negócio pode ter consequências consideráveis a longo prazo. Porque a definição de um processo é a impressão digital de um processo de negócio, é fundamental que não contenha erros graves. O processo deve também ser concebido de modo a que os tempos médios de conclusão e a capacidade necessária para os casos sejam mantidos tão pequenos quanto possível.

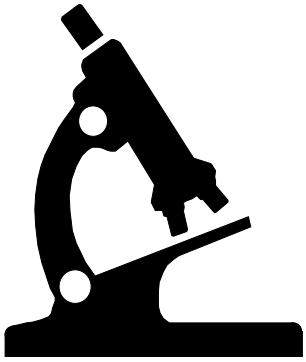


Figura 4.1. Técnicas de análises podem ser aplicadas para analisar *workflows* tanto qualitativamente como quantitativamente

Por exemplo, se duas tarefas podem ser realizadas em paralelo, em geral é sensato garantir que o processo o permita. Apesar de tudo, ao "paralelizar" tarefas, os tempos médios de conclusão geralmente podem ser reduzidos. Sendo a definição de um processo tão importante, é útil *analisá-la* completamente antes de a pôr em produção. Ao fazê-lo, diferenciamos entre a análise de (1) *aspectos qualitativos* e (2) de *aspectos quantitativos* dos *workflows*. O primeiro aspecto está relacionado com a *correcção lógica* do processo, isto é, a ausência de anomalias tais como "deadlocks" (quando um caso fica "bloqueado" e já não consegue prosseguir no processo) e "livelocks" (quando um caso fica preso num ciclo infinito). Os aspectos quantitativos dizem respeito principalmente ao

desempenho de um processo. A análise destes aspectos foca o estabelecimento de indicadores de desempenho, tais como o tempo médio de conclusão de um caso, o nível de serviço e a utilização de capacidade.

Neste capítulo, destacaremos um número de técnicas de análise que podem ser extremamente úteis no contexto da gestão de *workflow* (ver a figura 4.1). Introduzimos inicialmente uma técnica simples desenvolvida para ilustrar todos os estados que podem ser atingidos por um caso. Prestamos então atenção aos erros que podem ser cometidos ao desenhar a definição de um processo. Mostraremos que, com base na estrutura das redes de Petri, podemos decidir se a definição de um processo está correcta ou não. Na segunda parte deste capítulo iremos concentrar-nos na análise dos aspectos quantitativos. Usando alguns exemplos, iremos mostrar como melhorar o desempenho de processos existentes. Finalmente, iremos abordar o tema do planeamento da capacidade.

4.2 Análise de Alcançabilidade

Como vimos no capítulo 2, podemos definir um processo em termos de uma rede de Petri. A figura 4.2 ilustra uma rede deste género.

Uma rede de Petri e o seu estado inicial determinam quais os estados que são alcançáveis, e qual a ordem pela qual podem ser alcançados. (Como mencionámos no capítulo 2, o estado de uma rede de Petri corresponde à distribuição de testemunhos por lugares). Portanto, usamos redes de Petri para especificar o possível *comportamento* de um processo. Uma forma de descrever o comportamento é usar o *grafo de alcançabilidade.*

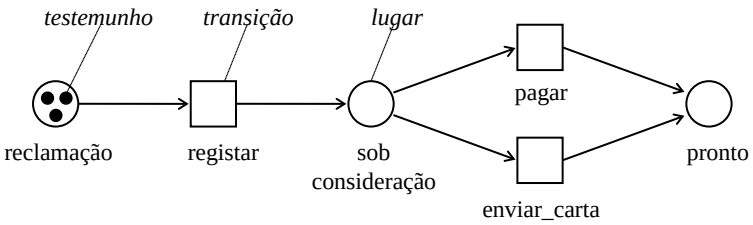


Figura 4.2. Uma rede de Petri clássica

Este é um grafo direccionado que consiste em nós e em arcos direccionados. Cada nó representa um estado que pode ser alcançado e cada arco uma possível mudança de estado. Para ilustrar estes conceitos, podemos observar a rede de Petri representada na figura 4.2. Os estados possíveis desta rede são indicados usando "triplos" *(a,b,c)* com *a* representando o número de testemunhos no lugar *reclamação*, *b* o número de testemunho em *sob_consideração*, e *c* o número em *pronto*. Definimos o estado inicial ilustrado com o elemento (3,0,0). O grafo de alcançabilidade que deriva deste estado inicial é ilustrado na figura 4.3.

Utilizando este grafo, conseguimos deduzir que existe um total de dez estados que podem ser alcançados. Cada nó representa um destes estados, mas cada estado não tem obrigatoriamente de ocorrer. O estado (1,2,0), por exemplo, é alcançado só se a transição *registar* for accionada pela segunda vez

quando o estado é (2,1,0). O número de arcos que saem de um nó indicam quantos estados subsequentes possíveis existem. Se existir mais do que um arco a sair, então o estado seguinte não está pré-determinado. Referimos esta situação como uma escolha *não determinística*. Se um nó não tem arcos à saída, então estamos perante um *estado final*. Este é um estado em que não são permitidas transições. O grafo de alcançabilidade na figura 4.3 mostra que a rede de Petri que começa com o estado (3,0,0) resulta sempre no estado final (0,0,3) depois de seis disparos.

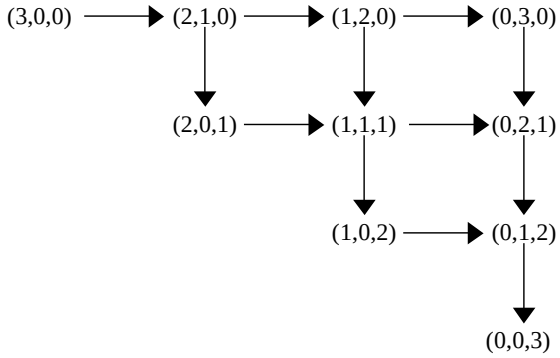


Figura 4.3. O grafo de alcançabilidade para a rede de Petri ilustrada na figura 4.2

Estamos prestando especial atenção ao grafo de alcançabilidade porque incorpora o comportamento do processo modelado. Desenhando o grafo de alcançabilidade para um número de casos podemos adquirir uma visão clara do funcionamento das redes de Petri. O facto é que, dado um diagrama como o da figura 4.2 (isto é, uma rede de Petri e o seu estado inicial), podemos compilar um grafo de alcançabilidade, o que mostra que as redes de Petri são uma forma não ambígua e precisa de descrição. Porque o funcionamento de uma rede de Petri é completamente formalizado, é também possível para um computador construir o grafo de alcançabilidade.

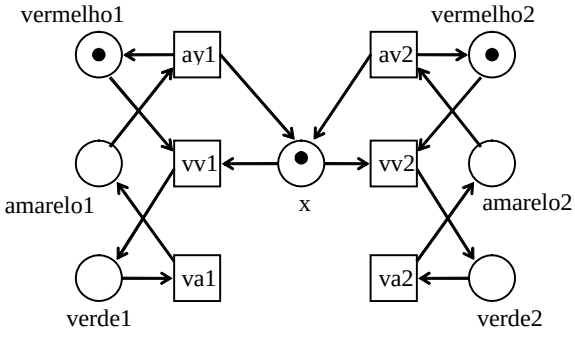


Figura 4.4. Dois grupos de semáforos

Como já vimos no segundo capítulo, podemos utilizar as redes de Petri para descrever processos com uma natureza repetitiva. Usamos a rede ilustrada na figura 4.4 para modelar dois semáforos localizados na intersecção de duas vias de sentido único. Ambos os semáforos operam de tal forma que um deles está sempre vermelho.

Quando ambos os semáforos estão a vermelho, existe um testemunho no lugar *x*. Logo que um dos semáforos muda para verde, o testemunho desaparece de *x* e o outro semáforo é bloqueado. Só quando ambos os semáforos se encontrarem outra vez a vermelho é que o outro semáforo muda para verde. Usando o grafo de alcançabilidade representado na figura 4.5 podemos verificar se os semáforos funcionam realmente de uma forma segura.

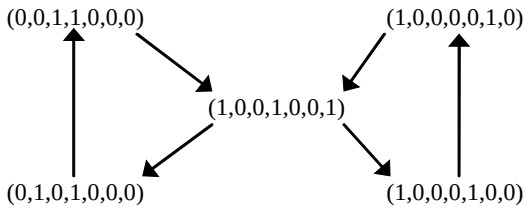


Figura 4.5. O grafo de alcançabilidade para a rede de Petri ilustrada na figura 4.4

Neste caso, cada estado possível é representado por sete elementos. As figuras mostram o número de testemunhos em *vermelho1*, *verde1*, *amarelo1*, *vermelho2*, *verde2*, *amarelo2*, e *x*, respectivamente. Uma inspecção do grafo de alcançabilidade mostra que os semáforos realmente operam com segurança: em qualquer estado possível pelo menos um dos semáforos está a vermelho. No entanto podemos ver que também é possível que o primeiro semáforo mude sempre para verde enquanto que o segundo permanece a vermelho. Podemos evitar esta situação assegurando que cada semáforo muda para verde alternadamente. A figura 4.6 mostra como esta situação pode ser modelada.

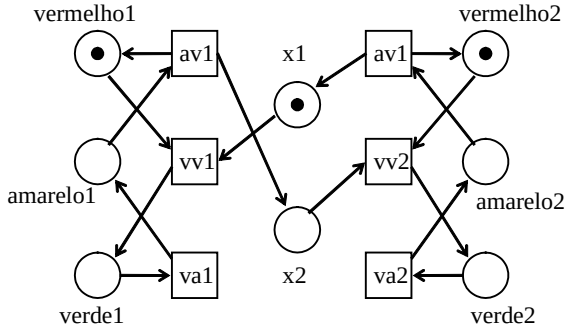


Figura 4.6. Os dois semáforos mudam agora para verde alternadamente

É fácil concluir que o grafo de alcançabilidade da figura 4.6 tem um total de seis estados. Da mesma forma que podemos verificar o correcto funcionamento dos semáforos utilizando o grafo de alcançabilidade, podemos utilizá-lo para verificar se um *workflow* está correcto. Antes de determinarmos se um processo está correcto, devemos dar atenção a um conjunto de erros típicos que podem ocorrer durante a definição de um processo.

4.3 Análise Estrutural

Antes da introdução de sistemas de informação avançados – tal como os sistemas de *workflow* – os processos de negócio tinham geralmente uma estrutura simples.

Isto deveu-se principalmente ao facto de um documento em papel estar sempre associado a um caso e, por isso, podia estar fisicamente apenas num lugar, num determinado momento. O documento agia como uma espécie de testemunho o qual se assegurava de que as tarefas eram realizadas sequencialmente. Como consequência dos vários desenvolvimentos das tecnologias de informação é agora possível organizar processos de forma completamente diferente. Com a utilização de bases de dados e redes de comunicação, a informação pode ser partilhada. Pelo facto de diferentes pessoas poderem trabalhar no mesmo caso ao mesmo tempo, deixa de ser necessário executarem-se as tarefas sequencialmente. Graças a "paralelização" dos processos de negócio podem ser conseguidos enormes reduções nos tempos médios de conclusão. No ambiente no qual o sistema de *workflow* funciona é frequentemente possível realizar as tarefas em paralelo, tanto quanto possível. Mas o uso de encaminhamento sequencial, paralelo, selectivo e iterativo no mesmo processo pode tornar difícil verificar se um processo está definido de forma correcta. Podemos mostrar isto usando o exemplo elementar ilustrado na figura 4.7.

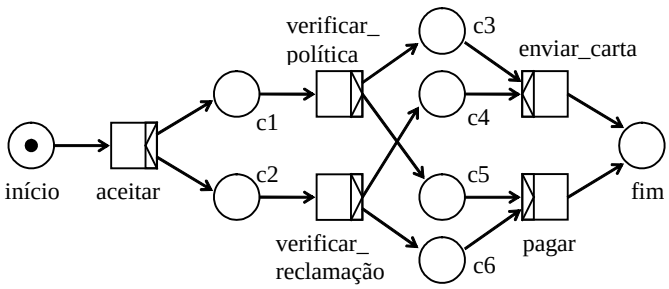
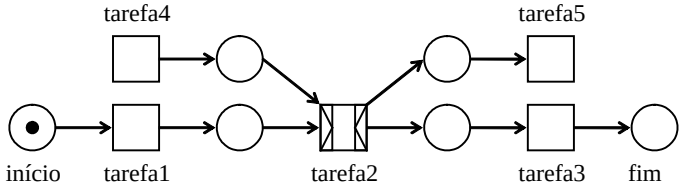


Figura 4.7. Exemplo de um processo incorrecto

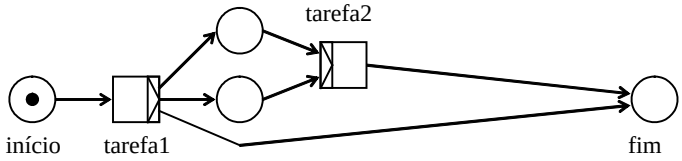
À primeira vista, parece tratar-se da definição de um processo melindroso, com duas tarefas de verificação executadas em paralelo, seguidas da aceitação de uma reclamação. Com base nestas verificações, ou uma carta de rejeição é enviada ou um pagamento é feito. No entanto, devido a uma má combinação de encaminhamentos paralelos e selectivos, os erros acumulam-se nesta definição de processo. Se *verificar_política* coloca um testemunho em *c5* e

verificar_reclamação em *c6*, o pagamento disparará. Este é o único cenário em que o caso é concluído com sucesso. Se *verificar_política* coloca um testemunho em *c3* e *verificar_reclamação* em *c4*, então *enviar_carta* disparará duas vezes. Como consequência dois testemunhos aparecem no lugar *fim*. Se *verificar_política* coloca um testemunho em *c3* e *verificar_reclamação* um testemunho em c6 então *enviar_carta* disparará somente uma vez, mas um testemunho permanecerá em *c6*. O mesmo acontece se *verificar_política* coloca um testemunho em *c5* e *verificar_reclamação* coloca um testemunho em *c4*.

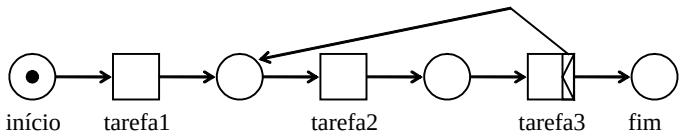
A figura 4.8 ilustra quatro situações que, como no exemplo anterior, podem resultar em processos incorrectos. Utilizando este exemplo podemos destacar um conjunto de erros comuns que ocorrem durante a definição de um processo:



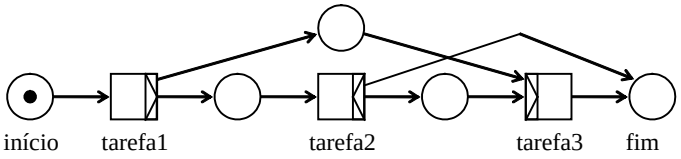
Situação A



Situação B



Situação C



Situação D

Figura 4.8. Quatro situações incorrectas

1. *Tarefas sem condições de entrada e/ou saída.* Quando uma tarefa não tem condições de entrada torna-se imprevisível quando é que pode ser executada. Quando uma tarefa não tem condições de saída, não contribui para a conclusão

de nenhum caso e por isso pode ser eliminada. A situação A, na figura 4.8, contém uma tarefa sem condições de entrada *(tarefa4)* e outra sem condição de saída *(tarefa5)*.

2. *Dead tasks (tarefas mortas)*: *são tarefas que nunca são executadas*. Obviamente que uma definição de processo que contém tarefas "mortas" é indesejável. Na situação B, a *tarefa2* nunca é executada; o mesmo acontece com a *tarefa3* na situação D.

3. *Deadlock (bloqueio): parar um caso antes da condição "fim" ser alcançada*. Se a *tarefa1* na situação B coloca um testemunho num dos dois lugares superiores, então o caso irá esperar indefinidamente pela *tarefa2*. Só se a *tarefa1* colocar directamente um testemunho no lugar *fim* é que o *deadlock* pode ser evitado. Na situação D, um testemunho pode ficar preso à espera da *tarefa3*.

4. *Livelock: aprisionar um caso num ciclo infinito*. Na situação C, todos os casos permanecem num ciclo infinito que consiste na *tarefa2* e na *tarefa3*. Assim, existe um encaminhamento iterativo que não tem oportunidade de saída.

5. *Actividades que ocorrem depois da condição "fim" ter sido alcançada*. Uma boa definição de processo tem um início (condição *início*) e um fim (condição *fim*) bem definidos. Quando a condição *fim* é alcançada, mais nenhuma tarefa deve ser realizada. Na situação C, as *tarefas2* e *tarefas3* irão disparar depois da condição *fim* ter sido alcançada. Desta forma, um número infinito de testemunhos irá alcançar o lugar *fim*. Esta é claramente uma situação indesejável.

6. *Testemunhos permanecem na definição do processo mesmo depois do caso estar concluído*. A partir do momento em que um testemunho aparece no lugar *fim*, todas as outras referências ao caso devem ter desaparecido. Na situação D, se o caso concluir em resultado do disparo da *tarefa2*, irá permanecer um testemunho num dos lugares anteriores à *tarefa3*.

Estes cenários demonstram que, sem qualquer conhecimento do conteúdo do processo que está sendo definido, conseguimos identificar um conjunto de erros típicos numa dada definição de processo. Estes erros estão associados ao encaminhamento de casos. A fim de computorizar a verificação de tais erros, necessitamos de uma noção mais precisa do que é um processo correcto.

4.3.1 Correcção

Nos restantes capítulos deste livro, exigimos que cada processo respeite os seguintes requisitos mínimos:

Um processo não deve conter tarefas desnecessárias e, cada caso submetido ao processo, deve ser concluído na totalidade. Além disso, não devem permanecer referências ao caso (isto é, testemunhos do caso) no processo.

Ao processo que cumpre estes requisitos mínimos, designamos de *correcto* (*sound*). Iremos formular a propriedade de *correcção* (*soundness*) de um processo usando a figura 4.9.

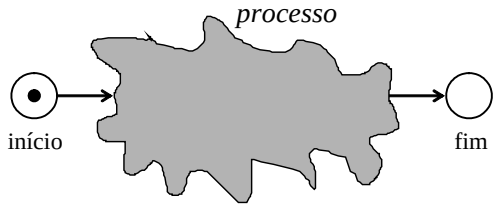


Figura 4.9. Um processo tem uma entrada e uma saída

Um *workflow* definido em termos de uma rede de Petri tem um único lugar de entrada *início* e um único lugar de saída *fim*. Tal rede de Petri só faz sentido se cada transição (tarefa) ou lugar (condição) estiver num caminho directo do lugar *início* para o lugar *fim*. Por outras palavras, não devem existir condições nem "tarefas soltas". De acordo com este requisito, cada tarefa (ou condição) pode ser alcançada desde o lugar de *início* seguindo um conjunto de arcos e o lugar *fim* é sempre alcançado por cada tarefa (ou condição) seguindo um conjunto de arcos. Uma transição que não esteja no caminho entre o lugar *início* e o lugar *fim* não contribui para uma conclusão bem sucedida do processo ou pode ser activada a qualquer momento. Nesta secção vamos apenas considerar as redes de Petri que satisfazem este requisito. Estas redes de Petri são designadas de redes de *workflow* (*workflow nets* – WF-nets).

Uma rede de *workflow* satisfaz alguns requisitos sintácticos. No entanto, podem existir redes de *workflow* que possuam anomalias, tais como *deadlocks* ou a incapacidade de terminar. Por isso, definimos que uma rede de *workflow* é correcta se, e só se, cumprir os seguintes três requisitos:

1. Para cada testemunho colocado no lugar *início*, um (e só um) testemunho aparece eventualmente no lugar *fim*;
2. Quando o testemunho aparece no lugar *fim*, todos os outros lugares estão vazios; e
3. Para cada transição (tarefa), é possível transitar de um estado inicial para um estado em que essa transição está pronta.

O primeiro requisito garante que todo o caso vai ser realizado com sucesso num determinado período de tempo. O segundo requisito garante que, quando um caso termina, nenhuma referência ao mesmo permanecerá no processo. Se combinarmos os dois primeiros requisitos, concluímos que – com base no estado ilustrado na figura 4.9 – existe apenas um estado final: isto é, precisamente um único testemunho no lugar *fim*. O último requisito exclui "tarefas mortas" (*deadtask*), isto é, cada tarefa pode ser – em princípio – executada.

A definição de correcção assume a noção de justiça, isto é, se uma tarefa pode potencialmente ser executada, então não é possível adiar a sua execução indefinidamente. Considere, por exemplo, o encaminhamento iterativo. Embora, em princípio, seja possível repetir parte de um processo infinitamente,

assumimos que esta iteração não viola necessariamente o requisito de correcção. De forma semelhante, assumimos que uma tarefa não pode depender indefinidamente de outras duas tarefas. Se não assumíssemos estas suposições, qualquer processo com encaminhamento iterativo ou selectivo não seria correcto.

Como podemos determinar se um dado processo é uma rede de *workflow* correcta? Esta determinação envolve os seguintes passos. Primeiro é necessário verificar se a rede de Petri que representa o processo é uma rede de *workflow*. Isto pode ser verificado examinando a estrutura do processo. Verificar se um processo está correcto é mais complicado. Podemos verificar os três requisitos de correcção usando um grafo de alcançabilidade, começando com o estado inicial, onde existe apenas um testemunho no lugar *início*. Para verificar este último requisito, verificamos se existe, para cada tarefa, uma transição no grafo de alcançabilidade que corresponde ao disparo dessa tarefa. Os dois primeiros requisitos são verificados determinando se o grafo de alcançabilidade tem apenas um estado final no qual existe precisamente um testemunho no lugar *fim*. Os requisitos para determinar se o processo está correcto podem então ser verificados automaticamente pela inspecção do grafo de alcançabilidade.

Existe, no entanto, dois inconvenientes ligados à esta aproximação. Em primeiro lugar, a construção do grafo de alcançabilidade para processos de grande escala pode envolver muito tempo computacional. É por isso quase impossível executar esta análise sem o auxílio de um computador. Em segundo lugar, o grafo de alcançabilidade fornece pouco suporte na reparação da definição de um processo que não seja correcto. Note-se que o grafo de alcançabilidade é infinito se testemunhos se puderem acumular num determinado lugar. É possível usar variantes do grafo de alcançabilidade, tal como o grafo de cobertura, que permite a detecção de tais comportamentos ilimitados (ver apêndice). Todavia, estas aproximações de "força bruta" podem consumir muito tempo e não fornecem bons diagnósticos.

Felizmente, existem técnicas disponíveis para as redes de Petri que não possuem estas desvantagens. Por agora, não temos oportunidade de discutir estas técnicas de forma detalhada. No entanto, iremos esboçar dois métodos alternativos para determinar se um processo está correcto. O primeiro método baseia-se no uso do computador; o segundo pode ser utilizado manualmente.

4.3.2 Método com recurso ao computador

O primeiro método para determinar a correcção traduz a propriedade de correcto para duas propriedades bem conhecidas, as quais foram investigadas durante décadas. A fim de analisar um processo definido em termos de uma rede de Petri, adicionamos uma transição adicional à rede: *t**. *t** tem *fim* como ponto de entrada e *início* como ponto de saída. A rede sem a transição *t** é designada de rede de *workflow*; a rede com a transição *t** é designada de rede com um curto-circuito (*short-circuited net*). Com esta adição, a correcção da rede de *workflow* corresponde a duas propriedades bem conhecidas: *vivacidade* (*liveness*) e *limitação* (*boundedness*) da rede com um curto-circuito. Uma rede de Petri está *viva* (*live*) quando é possível alcançar – para cada transição *t* e para todo o estado alcançável desde o *início* – um estado na qual a transição *t* está pronta. Numa rede de Petri viva, consequentemente, é possível activar todas as

transições um número arbitrário de vezes. Uma rede de Petri é limitada quando existe um limite superior para o número de testemunhos em cada lugar. Por outras palavras, não é possível que o número de testemunhos num lugar cresça indefinidamente se o processo começou no estado inicial. Os semáforos ilustrados nas figuras 4.4 e 4.6 são exemplo de processos vivos e limitados.

A vivacidade e limitação são propriedades que têm sido investigadas de forma intensiva durante os últimos trinta anos. Como resultado, algoritmos e ferramentas eficientes estão disponíveis para analisá-las. Um processo está correcto se a sua rede de Petri, com a transição t^* é viva e limitada. Determinar se um processo está correcto pode ser feito utilizando ferramentas *standard*. Para um número de subcategorias importantes – incluindo as chamadas *redes de Petri de escolha livre (free-choice Petri nets)* – a vivacidade e a limitação de uma rede podem ser verificadas em tempo polinomial. Graças aos muitos resultados conseguidos no campo da teoria das redes de Petri, a correcção de um processo pode então ser determinada de forma eficiente. Quando um processo não está correcto, podem ser gerados diagnósticos que indicam *o porquê* da situação.

Esta secção ilustra as muitas possibilidades de análise oferecidas pela representação de um processo usando redes de Petri. Para mais informação, aconselhamos ao leitor, o apêndice deste livro e a consulta da vasta documentação existente sobre as redes de Petri.

4.3.3 Método sem suporte computacional

A tradução de correcção para vivacidade e limitação permite a aplicação de técnicas eficientes de análise. Infelizmente, a tradução não é muito intuitiva e requer o suporte do computador para ser relevante. Consequentemente, propomos um método alternativo fácil de aplicar sem suporte computacional ou conhecimento teórico profundo. Adicionamos um requisito para uma "boa" rede de *workflows* para além de correcção: requereremos que as redes de *workflow* também sejam seguras (safe), o que significa que o número dos testemunhos em cada lugar nunca será maior a um, (isto significa que estão limitados pelo valor um). É muitas vezes fácil de verificar se uma rede é segura pela inspecção da sua estrutura. O método é baseado numa propriedade importante e intuitiva que é muito fácil de compreender:

Tendo duas redes de *workflow* seguras e correctas V e W, temos a tarefa t em V que tem precisamente um lugar de entrada e um de saída, então podemos substituir a tarefa t em V por W, obtendo assim novamente uma rede de *workflow* segura e correcta.

Na figura 4.10 esta substituição é ilustrada. Esta propriedade é intuitivamente nítida porque uma rede do *workflow* correcta comporta-se como uma transição: consome um testemunho do lugar de entrada, após um instante de tempo, produz um testemunho no lugar de saída. Por consequência, o ambiente não irá descobrir a substituição de t pelo W. A segurança das redes é requerida a fim de evitar a situação em que W tenha dois ou mais testemunhos activos em simultâneo, os quais poderão violar a correcção de W.

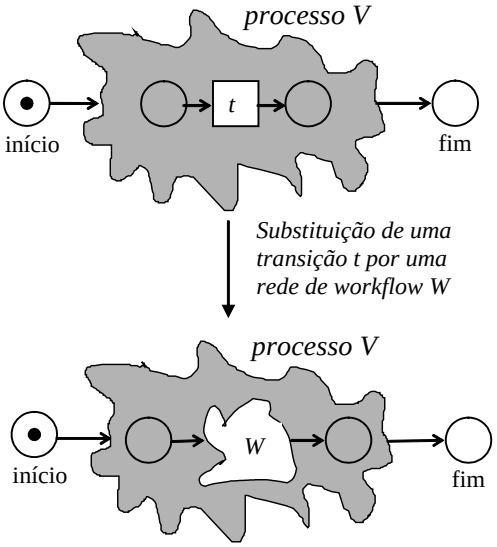


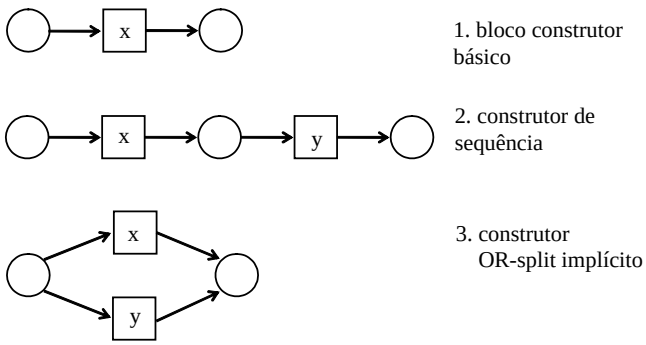
Figura 4.10. Se uma transição é substituída por uma rede de *workflow* correcta, então a rede de *workflow* resultante é também correcta (assumindo segurança).

Esta propriedade de substituição é demonstrada no apêndice. Aqui concentramo-nos na aplicação desta propriedade. A ideia principal é a seguinte:

Suponhamos que temos um conjunto de redes de *workflow* correctas e seguras, denominadas de "blocos construtores". Caso seja possível derivar a rede de *workflow* usando uma sequência de substituições das redes contidas neste conjunto de "blocos construtores", então provamos que a nossa rede é também correcta e segura.

Para ilustrar este método começamos com um pequeno conjunto de redes para o qual a correcção e a segurança são óbvias (ver figura 4.11). As redes de *workflow* correspondem às construções típicas introduzidas no capítulo 2. Existem, certamente, outros conjuntos de blocos construtores possíveis, mas este conjunto é já consideravelmente importante e significativo.

Vamos primeiro mostrar como podemos aplicar este método. Considere a rede de *workflow* ilustrada na figura 4.12.



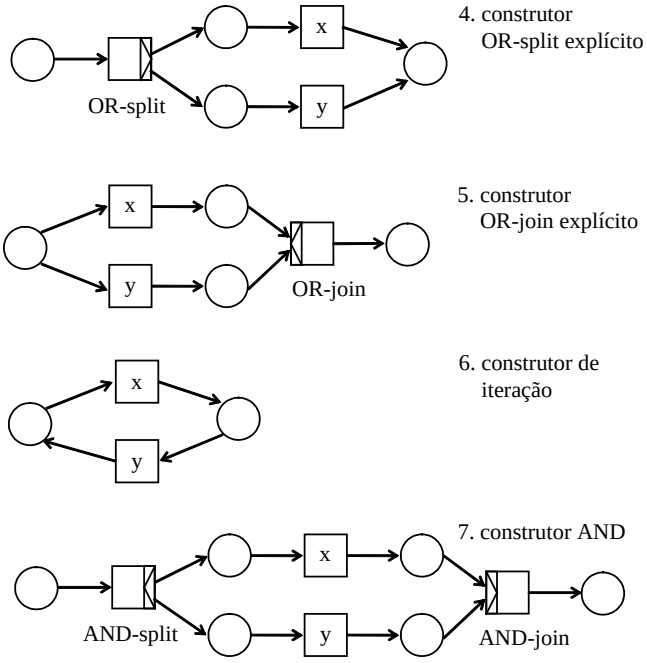


Figura 4.11. Redes correctas e seguras

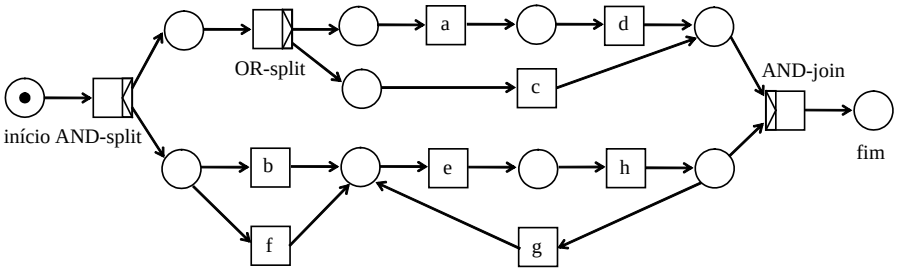


Figura 4.12. Um processo correcto e seguro

Para esta rede podemos encontrar a derivação apresentada nas figuras seguintes. O método começa com o bloco construtor básico ilustrados na figura 4.13.

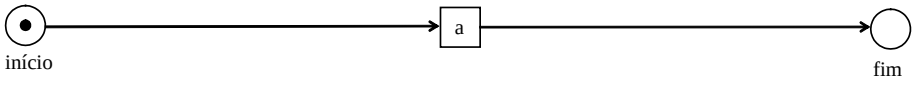


Figura 4.13. Antes de aplicar o construtor AND ao *a* (Etapa 1)

Na primeira etapa, o construtor AND é aplicado para colocar a tarefa *b* em paralelo com a tarefa *a*. A rede de *workflow* resultante está ilustrada na figura 4.14. Note-se que aplicamos simplesmente o construtor AND representado na figura 4.11 com $x = a$ e $y = b$.

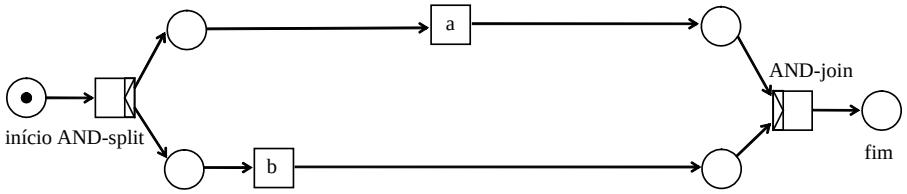


Figura 4.14. Aplicar o construtor explícito OR-split ao *a* (Etapa 2)

Na segunda etapa, o construtor OR-split explícito é aplicado a *a*, isto é, o padrão OR-split explícito ilustrado na figura 4.11 é aplicado com $x = a$ e $y = c$. A rede de *workflow* resultante está ilustrada na figura 4.15.

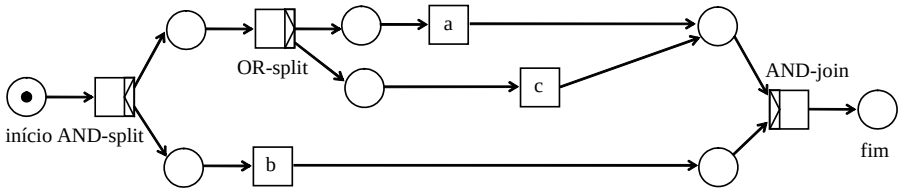


Figura 4.15. Antes de aplicar o construtor de sequência ao *a* (Etapa 3)

Na terceira etapa, aplicamos a construção de sequência: a tarefa *a* é seguida pela tarefa *d*.

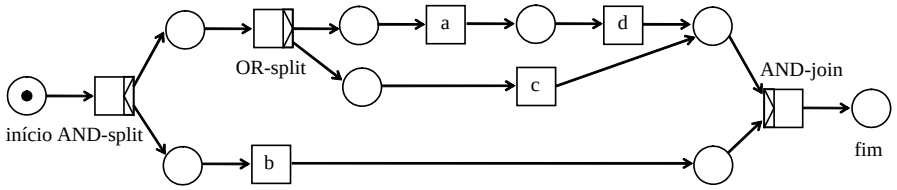


Figura 4.16. Antes de aplicar o construtor de sequência ao *b* (Etapa 4)

Depois, a construção de sequência é aplicada ao *b*.

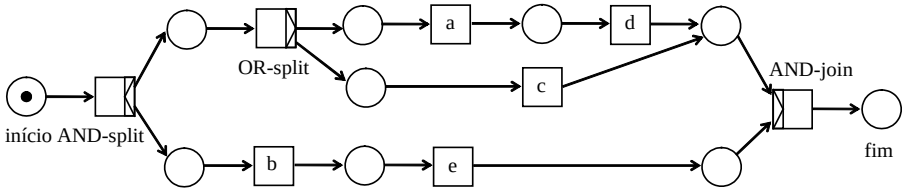


Figura 4.17. Aplicar o construtor OR-split implícito ao *b* (Etapa 5)

Na quinta etapa, um construtor OR-split implícito é aplicado a *b* com a adição da tarefa *f*.

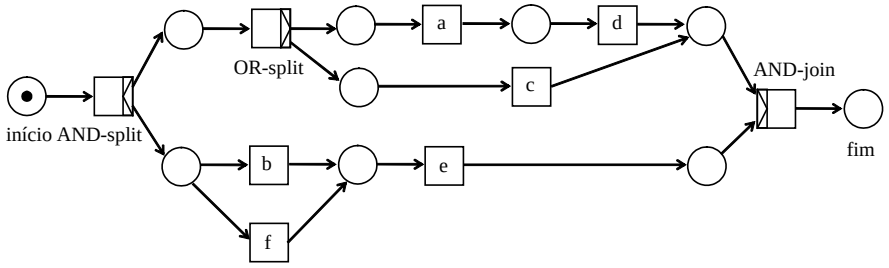


Figura 4.18. Antes da aplicação do construtor de iteração ao *e* (Etapa 6)

Depois é aplicada a construção de iteração a tarefa *e*. Em consequência, a tarefa *g* é adicionada à rede de *workflow*.

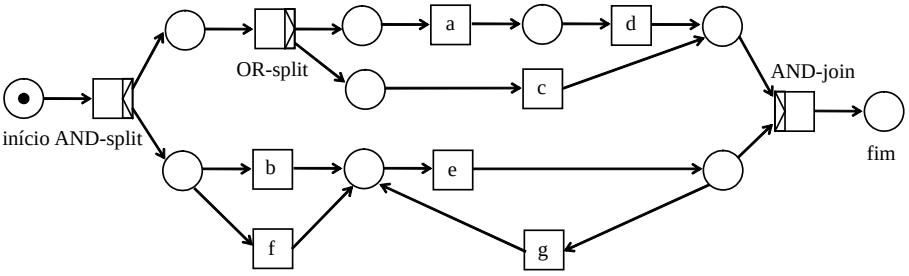


Figura 4.19. Antes de aplicar o construtor sequência ao *e* (Etapa 7)

Finalmente, o construtor de sequência é aplicado à tarefa *e*. A rede de *workflow* resultante, ilustrada na figura 4.20, é exactamente o processo que queríamos construir. Como apenas aplicamos os padrões de desenho ilustrados na figura 4.11 é garantido que a rede de *workflow* é correcta e segura.

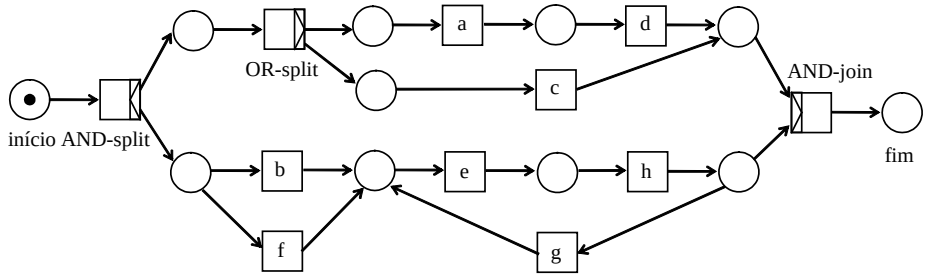


Figura 4.20. O processo completo

Como podemos ver pode existir mais de que uma derivação para uma rede particular. No exemplo poderíamos ter trocado as etapas 3 e 4. Nem todas as redes correctas e seguras têm uma derivação como está ilustrado no exemplo apresentado na figura 4.21.

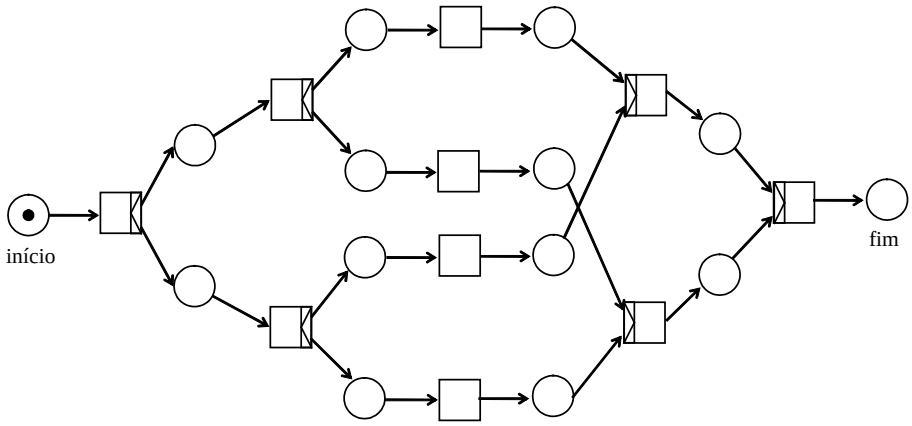


Figura 4.21. Um processo que não pode ser construído usando os construtores *standard* apresentados na figura 4.11

A razão pela qual não podemos encontrar uma derivação neste caso (figura 4.21) é o facto de que dois caminhos com origem num AND-split devem incidir no mesmo AND-join devido às regras de substituição apresentadas na figura 4.11. Este exemplo mostra que, no caso de não podermos encontrar uma derivação para uma rede de *workflow*, não é possível concluir se a rede não é correcta e segura. Na realidade, a rede de *workflow* ilustrada na figura 4.21 é correcta e segura, mas não é possível construir esta rede usando os blocos construtores *standard* apresentados na figura 4.11.

É de realçar que é sempre possível adicionar uma rede correcta e segura ao nosso conjunto de blocos construtores inicial, por isso também podemos adicionar a rede ilustrada na figura 4.21. Uma extensão particular das nossas regras de substituição é trivial: cada lugar (excluindo os lugares de início e fim) pode ser substituído por um lugar e uma tarefa no qual este lugar é a entrada,

assim como, o lugar de saída. Esta transformação está representada na figura 4.22.

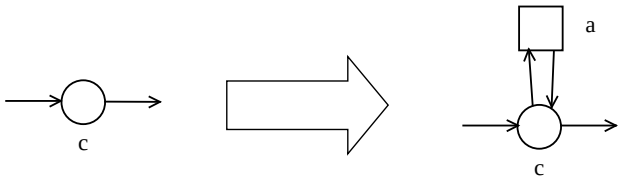


Figura 4.22. O construtor de ciclo

Suponha que encontramos uma derivação para uma rede e que temos de modificar a rede durante um processo de desenho. Caso as modificações consistam na substituição das tarefas por blocos construtores correctos e seguros, então não existe nenhum problema. Mas, suponhamos que temos de fazer uma outra modificação: será necessário encontrar uma nova derivação a partir do início? A resposta é não. Podemos sempre recuar na derivação e escolher uma outra sequência de etapas. Vamos ver como isso pode ser feito. Até ao momento, cada regra de substituição, substitui uma transição por duas transições com exactamente um lugar de entrada e um lugar de saída (os construtores ilustrados na figura 4.11). Em cada caso, o número de transições com uma entrada e uma saída aumenta de uma unidade. Quando substituímos uma transição por duas novas (com uma entrada e uma saída) então temos de dar à segunda (ou à primeira) transição adicionada um nome, único e exclusivo. Podemos caracterizar cada passo numa derivação por um triplo: a tarefa seleccionada, o bloco construtor utilizado, e o nome da nova tarefa. Na derivação ilustrada nas figuras 4.13 até 4.20 todas as tarefas têm um nome. Na tabela 4.1 representamos esta derivação de uma forma tabular.

Tabela 4.1. Cada etapa numa derivação

Etapa	Conjunto de tarefas	Tarefa seleccionada	Bloco usado	Nova tarefa
1	a	a	AND	b
2	a, b	a	OR-split explícito	c
3	a, b, c	a	sequência	d
4	a, b, c, d	b	sequência	e
5	a, b, c, d, e	b	OR-split implícito	f
6	a, b, c, d, e, f	e	iteracção	g
7	a, b, c, d, e, f, g	e	sequência	h

É fácil verificar que o resultado desta derivação é a rede com as tarefas {*a,b,c,d,e,f,g,h*} ilustradas na figura 4.20. Repare-se que não mencionamos tarefas adicionadas com finalidade de encaminhamento, isto é, os AND-split, os AND-join, e os OR-split são omitidos.

Suponhamos que queremos estender as redes de *workflow* representadas na figura 4.20 com uma tarefa adicional *x* de forma a obter a rede de *workflow* representada na figura 4.23.

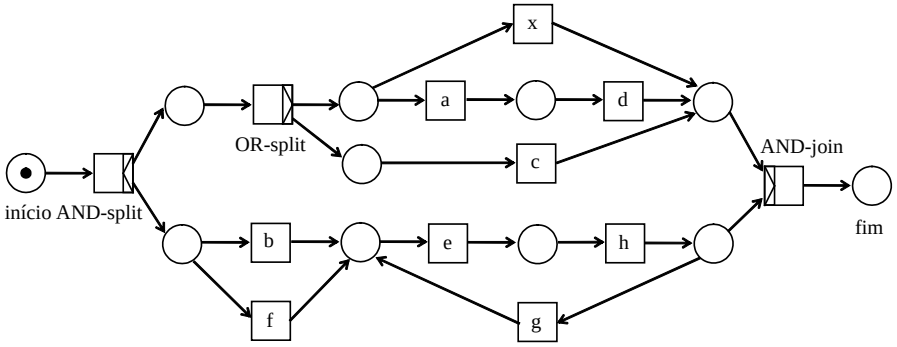


Figura 4.23. Um processo alternativo com uma tarefa adicional *x*

Repare-se que a tarefa *x* é adicionada pela introdução de um OR-split implícito. Como foi anteriormente mencionado, podemos usar a derivação anterior e simplesmente adicionar uma etapa entre 2 e 3 (etapa 2.5). Após esta modificação podemos continuar a derivação, da qual resulta a rede com as tarefas {*a,b,c,d,e,f,g,h,x*} ilustradas na figura 4.23. A tabela 4.2 representa esta derivação. Usando esta técnica simples podemos construir um grande conjunto de redes de *workflow* correctas e seguras.

Tabela 4.2. Cada etapa numa derivação (com 2.5)

Etapa	Conjunto de tarefas	Tarefa Seleccionada	Bloco usado	Nova tarefa
1	a	a	AND	b
2	a, b	a	OR-Split explícito	c
2.5	a, b, c	a	OR-split implícito	x
3	a, b, c, x	a	sequência	d
4	a, b, c, d, x	b	sequência	e
5	a, b, c, d, e, x	b	OR-split implícito	f
6	a, b, c, d, e, f, x	e	iteracção	g
7	a, b, c, d, e, f, g, x	e	sequência	h

4.4 Análise de Desempenho

Da mesma forma como estamos interessados em verificar se um modelo de *workflow* está correcto, também estamos interessados no seu *desempenho*. Por isso, mencionamos aspectos quantitativos relacionados com os tempos de conclusão dos casos, o número de casos que podem ser processados por unidade de tempo, a utilização dos funcionários e a percentagem de casos que podem ser completados num intervalo de tempo. Para obter dados sobre o desempenho de um modelo de *workflow*, várias técnicas de análise podem ser usadas. As três técnicas mais usuais são as seguintes:

1. *Análise de Markov*. Com base num *workflow*, é possível gerar uma cadeia de Markov automaticamente. A cadeia pode ser usada para analisar aspectos particulares de um *workflow*. Tal cadeia contém os estados possíveis de um caso e as probabilidades associadas às transições entre eles. De facto, uma cadeia de Markov é um grafo de alcançabilidade com probabilidades associadas às transições. Estas probabilidades são determinadas com base nas propriedades previstas ou medidas de cada tipo de caso. Várias propriedades podem ser estabelecidas usando uma cadeia de Markov, por exemplo, a possibilidade de um determinado caso escolher um caminho específico num processo. Ao expandir as cadeias de Markov com aspectos de custo e de tempo, um conjunto de indicadores de desempenho pode ser gerado. A desvantagem desta aproximação é que nem todos os aspectos podem ser incorporados na análise. A análise de cadeias de Markov pode consumir muito tempo (ou ser mesmo intratável – *intractable*).

2. *Teoria das filas*. A teoria das filas é usada para a análise de sistemas nos quais a ênfase recai nos indicadores de desempenho tais como: tempos de espera, tempos de conclusão e utilização de capacidade. Consequentemente, é uma maneira lógica de analisar workflows. Num *workflow*, podem existir filas de casos à espera de recursos que estão ocupados e não podem processar imediatamente uma afluência particular de casos. Se estivermos interessados na formação de uma única fila para um número de recursos iguais, então podemo-nos confinar a um sistema que consiste numa fila única. Existem muitos resultados disponíveis para a análise de filas únicas que são geralmente simples de aplicar. Caso pretendamos avaliar todo um *workflow*, então necessitamos de considerar uma rede de filas. Para redes de filas, algumas perguntas podem ser respondidas usando métodos matemáticos. Infelizmente, muitas das suposições usadas na teoria das filas são inválidas para processos de *workflow*. Por exemplo, perante o encaminhamento paralelo é muitas vezes impossível aplicar os resultados obtidos da teoria das filas.

3. *Simulação*. A simulação é uma técnica muito flexível de análise. É quase sempre possível analisar um *workflow* por simulação. De facto, esta técnica resume-se ao seguimento de um caminho no grafo de alcançabilidade. Assim sendo, as escolhas são feitas a partir de várias distribuições probabilísticas. A simulação não é mais do que a execução repetida de um processo com a ajuda de um computador, é uma técnica que está acessível a pessoas sem bases matemáticas. Consequentemente, a simulação resulta numa melhor introspecção nas operações do processo a ser modelado. Porque a maioria das ferramentas de simulação oferecem uma opção de animação, o *workflow* pode ser monitorizado graficamente. Além disso, a simulação pode ser usada para responder a um grande leque de questões. Também é fácil estender um modelo de simulação com novos aspectos (por exemplo, falhas). Contudo, a realização e a análise de um modelo para uma simulação detalhada podem ser exigente a nível de tempo. Além disso, um processamento cuidadoso dos resultados da simulação requer conhecimento estatístico aprofundado.

Neste livro, a simulação é a principal técnica de análise. A razão pela qual nos confinamos apenas à técnica de análise é que a simulação é geralmente a única ferramenta suportada por sistemas de gestão de *workflow*. Quando

examinamos as técnicas de análise usadas no BPR, vemos uma vez mais que a simulação é, normalmente, a única ferramenta disponível para realizar análises quantitativas. Para ilustrar o uso de uma técnica de análise como a simulação, utilizamos a definição do processo ilustrado na figura 4.24.

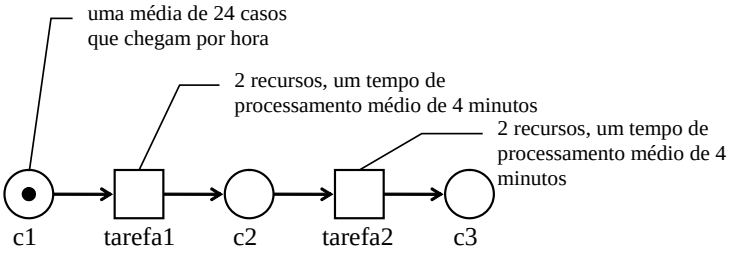


Figura 4.24. Situação 1

Como está ilustrado na figura 4.24, o processo consiste em duas tarefas a serem executadas sequencialmente. O número médio de novos casos que chegam ao processo por hora é de 24. O tempo médio entre duas chegadas sucessivas é, consequentemente, 2.5 minutos. O tempo médio requerido para realizar ambas as tarefas, *tarefa1* e *tarefa2*, é de 4 minutos para cada uma. Para cada tarefa, dois recursos são dedicados exclusivamente para processar os itens de trabalho. Estes, por conseguinte, são recursos altamente inflexíveis, os quais podem trabalhar somente para uma tarefa. Com base nas figuras anteriores, podemos calcular o nível médio de utilização dos recursos, isto é, o número de casos que chegam por unidade de tempo dividido pelo número de casos que podem ser servidos por unidade de tempo. Neste caso particular é de 80 por cento: em média um recurso gasta 80 por cento do seu tempo a trabalhar numa tarefa para um caso particular. O recurso está inactivo para os restantes 20 por cento do tempo.

Podemos agora perguntar, qual o tempo médio de conclusão para um caso. Para responder a esta questão, necessitamos de saber mais sobre o padrão de chegada de novos casos e sobre o tempo de processamento. Por ser mais conveniente, assumimos que os tempos entre chegadas seguem uma distribuição exponencial negativa. Considerando esta hipótese, podemos determinar, usando a simulação ou a teoria das filas, que o tempo de conclusão médio é aproximadamente de 22.2 minutos. Por outras palavras, leva uma média de 22.2 minutos para que um caso transite do lugar *c1* para o lugar *c3*. Mas destes 22.2 minutos, somente uma média de 8 minutos é realmente gasta no trabalho sobre o caso. Os restantes 14.2 minutos correspondem a tempo de espera. Consequentemente, neste caso, o tempo médio de espera é, realmente, mais longo do que o tempo de processamento. De facto, esta é realmente uma situação presente em muitos cenários da vida real. Considere-se, por exemplo, o tempo gasto na espera de uma consulta médica. Em muitos processos administrativos, os cenários podem tornar-se mais graves: os tempos actuais de processamento são uma pequena fracção do tempo total de conclusão.

Como indicado numa das directrizes para o desenvolvimento de *workflows*, é sensato – quando possível – executar tarefas em paralelo. A figura 4.25 mostra o

processo que poderia ser usado se fosse possível realizar simultaneamente as duas tarefas para cada caso.

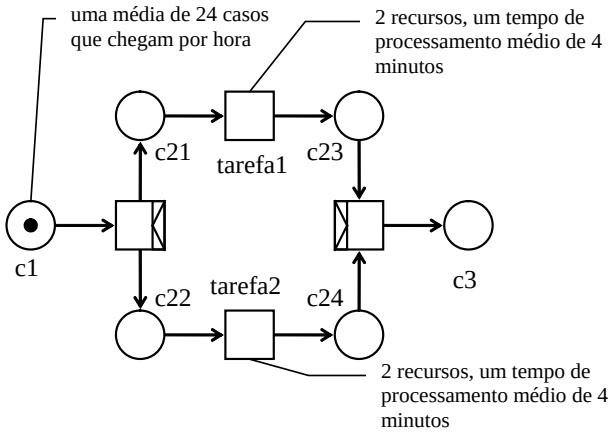


Figura 4.25. Situação 2

Nesta situação, o nível médio de utilização dos recursos mantém-se nos 80% – afinal, o fornecimento de casos e o tempo médio de processamento não mudaram. Contudo, o tempo médio de conclusão pode ser reduzido significativamente desta forma. Usando a simulação, podemos demonstrar que o tempo médio de conclusão é agora aproximadamente de 15 minutos. Ao executar tarefas em paralelo podemos atingir uma redução considerável do tempo de conclusão dos casos com os mesmos recursos.

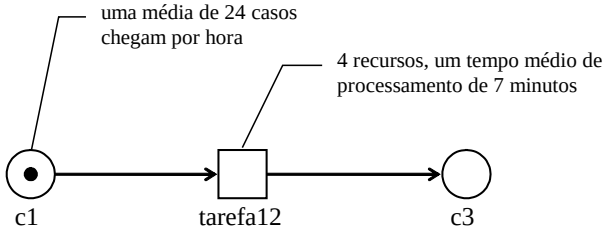


Figura 4.26. Situação 3

Pode às vezes ser vantajoso combinar duas tarefas numa tarefa maior. A figura 4.26 ilustra um processo em que a *tarefa1* e a *tarefa2* foram fundidas numa única *tarefa2*. O tempo médio de processamento para esta nova tarefa é de 7 minutos. Consequentemente, assumimos que leva menos de 1 minuto para realizar a tarefa combinada do que para realizar as duas tarefas originais. Esta redução é explicada pela eliminação do tempo de "aquecimento". Em consequência do tempo de processamento ser mais curto, o nível médio de utilização da capacidade dos recursos caiu para os 70 por cento. Além disso, o tempo de conclusão baixou dramaticamente para uma média de 9.5 minutos.

Assim, para cada caso há agora um tempo médio de espera de 2.5 minutos. Comparado com o tempo original médio de espera de 14.2 minutos, observamos uma melhoria considerável que é principalmente justificada pelo aumento da flexibilidade dos recursos. A nova *tarefa12* pode ser executada por cada um dos 4 recursos. Em contraste com a situação anterior, cada um dos recursos encontra-se ocupado enquanto existir um caso para ser executado.

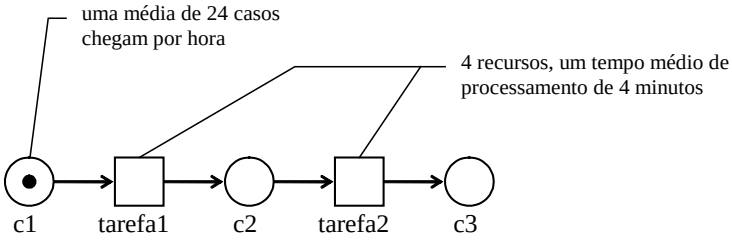


Figura 4.27. Situação 4

Para ilustrar a influência positiva da flexibilização dos recursos, considere o processo original representado na figura 4.27. Neste processo as duas tarefas continuam a ser realizadas sequencialmente. Contudo, neste caso os recursos não estão associados a uma tarefa específica: cada um pode executar a *tarefa1* e a *tarefa2*. Como resultado, o tempo médio de conclusão é somente de 14.0 minutos. Comparado com a situação original, o tempo médio de espera decresce de 14.2 para 6 minutos.

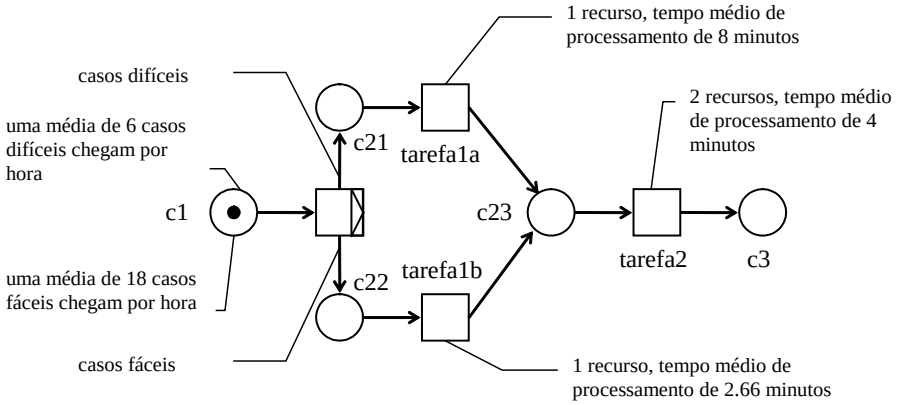


Figura 4.28. Situação 5

Até aqui, assumimos que os casos eram indistinguíveis entre si. Por outras palavras, nós não sabemos se o processamento de um caso particular demorará pouco ou muito tempo. A figura 4.28 representa a situação na qual podemos diferenciar entre casos "fáceis" e casos "difíceis". Executar a *tarefa1* para um caso fácil leva uma média de 2.66 minutos, enquanto que para um caso difícil leva uma média de 8 minutos. Em média, 25 por cento dos casos são classificados como difíceis, 75 por cento como fáceis. Na figura 4.28 tentamos

fazer uso dessa informação. Um recurso especial foi atribuído para executar a *tarefa1* para os casos difíceis. Adicionalmente, existe também um recurso especial para executar a *tarefa1* para casos fáceis. A ideia é que o tempo médio de conclusão pode ser reduzido pela separação dos dois fluxos. Este é um princípio conhecido por *triagem* (triagem é um termo que já existia muito antes da ascensão do BPR e do WFM. É também usado para descrever a selecção e a prioridade de acidentados de guerra ou desastres de acordo com a sua natureza e a severidade dos seus ferimentos.) Neste caso, porém, os resultados são desastrosos: o tempo médio de conclusão aumenta em pelo menos 31.1 minutos. Assim, existe uma degradação considerável do desempenho.

Existem situações em que a triagem pode ter um efeito benéfico. Considere, por exemplo, os cestos de compras na caixa de pagamento de um supermercado. Existem duas circunstâncias em que a triagem pode ser vantajosa: (1) quando a reserva de recursos especializados reduz o tempo médio de processamento e (2) quando os pequenos clientes não têm de esperar pelos grandes clientes para serem processados, o que reduz o tempo médio total de espera. A razão pela qual a triagem tem um efeito negativo na figura 4.28 é porque a flexibilidade dos recursos é reduzida. Por exemplo, somente um recurso pode executar a *tarefa1* para um caso fácil. Este exemplo mostra que uma análise quantitativa completa é frequentemente requerida para alcançar um bom desenho de um *workflow*.

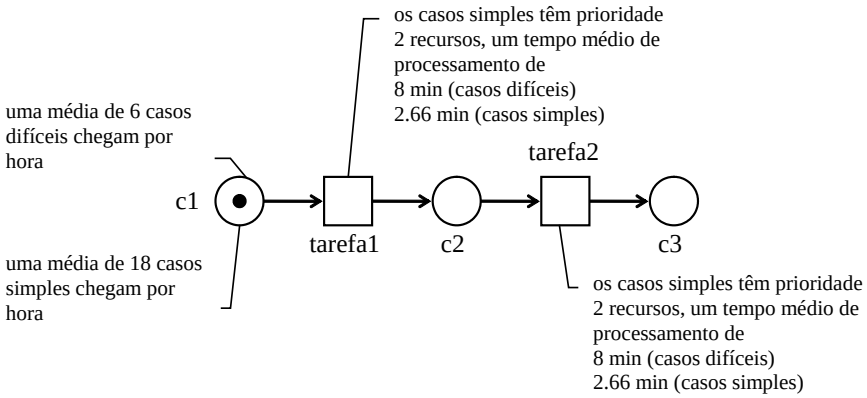


Figura 4.29. Situação 6

situação	descrição	tempo médio de conclusão	tempo médio de processamento	tempo médio de espera
situação 1	sequencial	22.2	8.0	14.2
situação 2	paralelo	15	4	11
situação 3	composição	9.5	7.0	2.5
situação 4	flexibilização	14.0	8.0	6.0
situação 5	triagem	31.1	8.0	23.1
situação 6	prioridades	14.0	8.0	6.0

Figura 4.30. Um sumário do desempenho nas seis situações descritas

A introdução da triagem num supermercado (caixa expresso) geralmente reduz o tempo total de conclusão, porque os clientes com somente alguns artigos não têm de esperar atrás dos clientes com muitos artigos. De facto, a triagem opera neste caso como uma regra de prioridade. Geralmente, a triagem induz a curtos tempos de conclusão quando os casos fáceis são realmente tratados mais cedo do que os casos difíceis. Se este não for o caso, implica maiores tempos de conclusão. Contudo, podemos aplicar uma regra de prioridade sem usar a triagem (por outras palavras, sem introduzir uma fila especial). A figura 4.29 ilustra uma situação em que para cada tarefa os casos fáceis (aqueles com um tempo médio de processamento de 2.66 minutos) têm prioridade sobre os difíceis (aqueles com um tempo médio de processamento de 8 minutos). Com a ajuda de técnicas de simulação podemos demonstrar que o tempo médio de conclusão é de aproximadamente 14 minutos. Assim, as regras de prioridade podem também dar o seu contributo economizando os tempos de conclusão. A figura 4.30 apresenta todas as situações sob a forma de um sumário.

Demonstramos, assim, que podemos usar uma técnica de análise como a simulação para suportar o desenho de um *workflow*. Dependendo do desenho, vimos que o tempo médio de espera para um caso pode variar de 2.5 minutos (situação 3) para mais de 23 minutos (situação 5). O desenho preferível depende das circunstâncias. Existe, no entanto, três directrizes que se aplicam à maioria das situações.

1. *Quando possível, executar tarefas em paralelo*. A implementação de processamento em paralelo resulta geralmente em tempos de conclusão mais curtos.

2. *Manter a flexibilidade elevada dos recursos*. Assegurar que os recursos possam executar tantas tarefas quanto possível. O uso de recursos flexíveis resulta em níveis mais elevados de utilização de recursos e de tempos de conclusão mais curtos.

3. *Quando possível, tratar dos casos em função do tempo de processamento*. Em geral, é sensato e vantajoso dar prioridade aos casos que têm um curto tempo de processamento sobre aqueles com maior tempo de processamento. Isto pode ser feito usando regras de prioridade ou de triagem.

Estas directrizes ilustram o facto de que existem semelhanças consideráveis entre a estrutura e gestão logística e os sistemas de produção. De facto, um sistema de *workflow* é um sistema de gestão logística. Consequentemente, é importante que ao projectar *workflows* se apliquem os princípios, os métodos e as técnicas que foram desenvolvidas para a estruturação, gestão, controle logístico e sistemas de produção.

4.5 Planeamento de Capacidade

Até aqui temos assumido que o número de recursos em cada classe de recursos é fixo. Na prática, com é evidente, isso não acontece. Os funcionários podem adoecer, ir de ferias, ou abandonar a empresa. O número de

funcionários pode também variar de acordo com factores sazonais. Considere, por exemplo, as vendas de seguros de viagens que são claramente sujeitas a influências sazonais. Estes factores precisam de ser considerados quando é feita a distribuição dos funcionários. Em certas indústrias também observamos que a geração de novos casos segue um padrão semanal. Assim, o *plano de capacidade* é sempre baseado numa *capacidade requerida* particular. O plano mostra que recursos e de que tipo são necessários para cada período. A capacidade de planeamento pode ser realizada tanto a curto como a longo prazo. No curto prazo, factores como licenças médicas, pequenas flutuações nas previsões de trabalho, feriados, trabalho extraordinário e o contrato de pessoal temporário têm um papel importante. No longo prazo, previsões de procura, influência sazonal, aquisição de maquinaria e recrutamento de pessoal são os factores a ter em conta.

Se tivermos uma previsão da geração de novos casos é fácil calcular os requisitos a nível da capacidade. Para ilustrar este facto, iremos usar uma variante do processo *tratar reclamações* introduzido no capítulo anterior. A figura 4.31 mostra o tempo médio de processamento para cada tarefa.

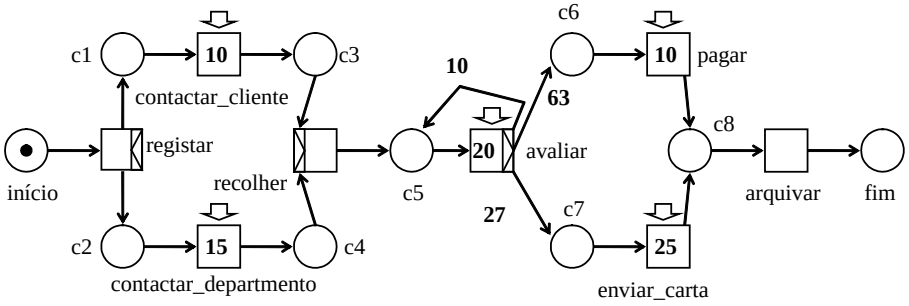


Figura 4.31. O processo tratar reclamações, com a indicação do tempo médio de processamento por tarefa

Tarefa	Número médio por dia	Tempo médio de processamento	Número médio de minutos
registar	50	0	0
contactar_cliente	50	10	500
contactar_dept.	50	15	750
recolher	50	0	0
avaliar	56	20	1111
pagar	35	10	350
enviar_carta	15	25	375
arquivar	50	0	0

Figura 4.32. A capacidade requerida por tarefa

É assumido que o tempo para executar as tarefas que não requerem recursos é insignificante. Para as outras, o tempo médio de processamento é expresso em minutos. Por exemplo, a tarefa *avaliar* demora em média 20 minutos. Em geral,

63% dos casos foram avaliados positivamente no fim desta tarefa e 27% negativamente. Nos restantes 10% dos casos é requerida uma avaliação adicional. Note-se que a tarefa *avaliar* pode ser executada um número arbitrário de vezes. O número médio de vezes que a avaliação é executada, por reclamação, é de 1.111 (ver secção 4.5.1). Eventualmente, 70% são avaliadas positivamente e 30% negativamente. Se assumirmos que 50 novos casos chegam cada dia, podemos calcular a capacidade requerida para cada tarefa. A figura 4.32 mostra que a tarefa *avaliar* requer a capacidade máxima.

Um caso é avaliado em média 1.111 vezes, porque 10% deles requerem uma segunda avaliação. Então, para uma entrada de 50 casos é requerida uma média de aproximadamente 56 avaliações. A capacidade requerida por tarefa é fácil de calcular nesta situação. Em processos mais extensos, com um grande número de iterações (ciclos), isto pode ser bastante mais complicado. Felizmente, com base na definição de um processo, é possível gerar automaticamente uma cadeia de Markov para calcular a capacidade requerida para cada tarefa.

Com base na capacidade requerida por tarefa podemos calcular a capacidade requerida para cada classe de recursos. Afinal de contas, sabemos a classe de recursos da qual um pedido de recurso virá. Como foi mencionado no capítulo anterior, existem 4 classes de recursos, neste caso: *Funcionário*, *Assessor*, *Reclamações* e *Finanças*. Cada recurso pertence a *Reclamações* ou a *Finanças*, mas não a ambos. Cada recurso que pertence à classe *Assessor* é automaticamente um membro da classe de recurso *Funcionário*. A tarefa *pagar* é a única que requer um recurso da classe de recurso *Finanças*. As outras tarefas requerem sempre um recurso da classe de recurso *Reclamações*. Além disso, a tarefa *avaliar* é a única que requer um recurso da classe de recurso *Assessor*. Com base nesta informação, a figura 4.33 mostra a capacidade requerida por classe de recurso.

Classe de recurso	Número médio de minutos	Número de recursos a 80% da capacidade	Número de recursos a 60% da capacidade
Funcionário	1975	5.14	6.86
Assessor	1111	2.90	3.86
Reclamações	2736	7.13	9.50
Finanças	350	0.91	1.22

Figura 4.33. A capacidade requerida por classe de recurso

A figura 4.33 mostra também o número de recursos requeridos a dois níveis particulares de capacidade de utilização. Quando esta é de 80%, o departamento de reclamações requer 8 pessoas, das quais, pelo menos 3 devem ser assessores. Porque existe uma sobreposição de classes de recurso, devemos interpretar os valores na figura 4.33 cuidadosamente. Por exemplo, todo o recurso na classe de recurso *Assessor*, também pertence à classe de recurso *Funcionário*. Contudo, os valores na fila para a categoria *Funcionário* referem-se apenas aos funcionários que não trabalham como assessores. Se compararmos os números da figura 4.33 com os recursos especificados no capítulo anterior, vemos que o departamento de reclamações está com falta de pessoal para um fluxo de 50 casos por dia. Por outro lado, o departamento de finanças tem um excesso de capacidade.

4.5.1 Método para calcular a capacidade requerida

Para a figura 4.31 é simples calcular a capacidade requerida listada nas figuras 4.32 e 4.33, para processos de *workflow* complexos isto pode ser mais complicado. Por isso, fornecemos directrizes mais concretas. Para determinar a capacidade requerida é importante saber o número médio de vezes que cada tarefa é executada. Na figura 4.31 as tarefas *registar*, *contactar_cliente*, *contactar_departamento*, *pagar* e *arquivar* são executadas precisamente uma vez. A tarefa *pagar* é executada 0.7 vezes, a tarefa *enviar_carta* é executada 0.3 vezes e a tarefa *avaliar* é executada em média 1.111 vezes. Como calcular o número médio de vezes que cada tarefa é executada? Uma forma é construir uma cadeia de Markov que é isomórfica com o grafo de alcançabilidade e adicionar as funções de custo apropriadas. A desvantagem desta aproximação é que a construção de uma cadeia de Markov requer um suporte informático e pode levar muito tempo. Também existe uma aproximação mais pragmática baseada em padrões de desenho (*design patterns*) descritos na figura 4.11. Estes padrões podem ser usados para construir redes de *workflow* seguras e correctas. Contudo, como mostra a figura 4.34, os padrões podem também ser usados para determinar o número médio de vezes que cada tarefa é executada.

Comparando com a figura 4.11, os padrões de desenho da figura 4.34 foram estendidos com números. Suponhamos que a tarefa x é executada N vezes na situação original, isto é, antes de aplicar o padrão. Se a construção sequencial é usada, então tanto a tarefa x como a tarefa y são executadas N vezes na nova situação. Se uma das três construções OR é aplicada, então x é executada αN vezes e y é executada $(1\text{-}\alpha)N$ vezes (em média). Note-se que α é a probabilidade de x ser executada na nova situação. Se a construção AND é utilizada, então tanto a tarefa x como a tarefa y são executadas N vezes na nova situação. A construção iterativa é um pouco mais complicada. Consideremos a como sendo a probabilidade de que depois de processado x seja necessária uma nova iteração. Podemos calcular que na nova situação, x é executada $N/(1\text{-}\alpha)$ vezes e que y é executada $\alpha N/(1\text{-}\alpha)$ vezes. Para entender estes valores considere a construção iterativa na figura 4.34. Sendo v o número esperado de vezes que x é executada para um caso inicial no lugar p, então a equação seguinte deve ser válida: $v = 1 + \alpha v$, já que acontece uma vez e com a probabilidade α de retornamos para o lugar inicial. Resolvendo esta equação obtemos $v = 1/(1\text{-}\alpha)$. A tarefa y é executada $v\text{-}1 = \alpha(1\text{-}\alpha)$ vezes. Por conseguinte, se o lugar inicial é marcado N vezes, x é executada $N/(1\text{-}\alpha)$ vezes e y é executada $\alpha N(1\text{-}\alpha)$ vezes.

Note-se que a rede de *workflow* ilustrada na figura 4.31 não pode ser construída usando os padrões de desenho apresentados na figura 4.34. O padrão de construção iterativo não pode ser usado para fazer o ciclo envolvendo *c5* e *avaliar*. No entanto, uma construção iterativa similar pode ser adicionada à lista de construções apresentada na figura 4.34. Se α é a probabilidade de *avaliar* ser executada novamente, então o número total de vezes que *avaliar* é executada é igual a $N/(1\text{-}\alpha)$.

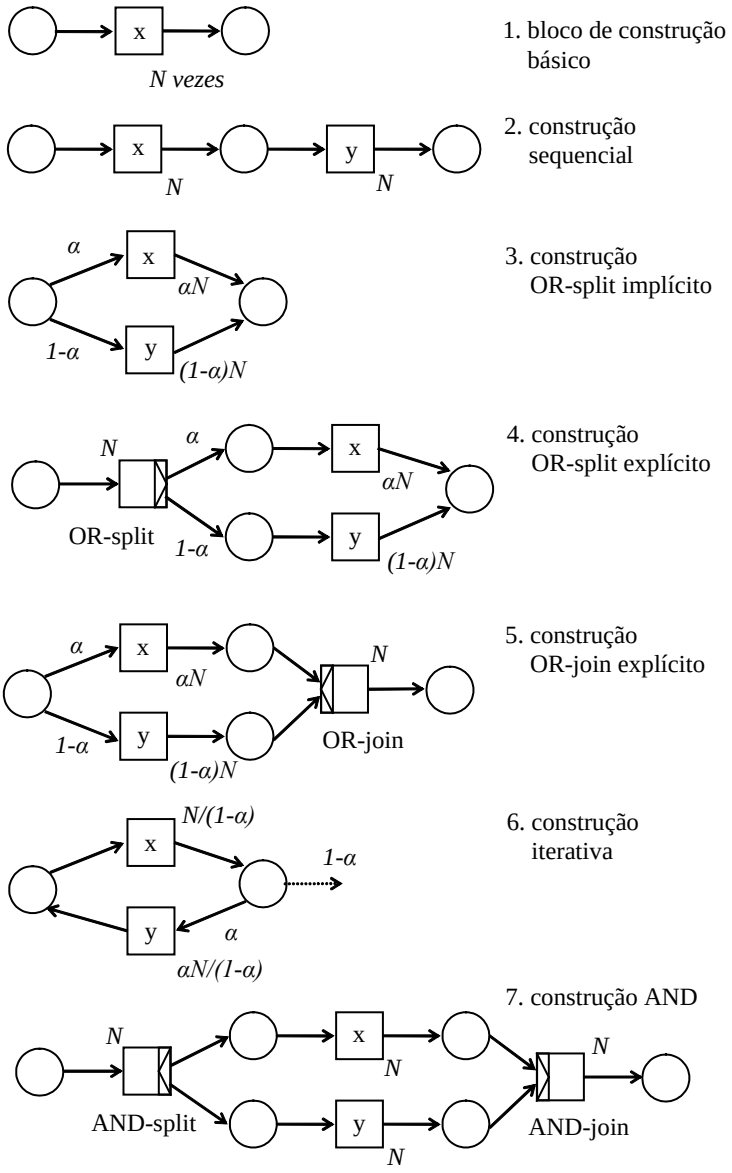


Figura 4.34. Número de vezes que cada tarefa é executada relativamente ao número de vezes que a tarefa *x* é executada na situação original

Se o número médio de novos casos por unidade de tempo e o número médio que cada tarefa demora a executar um caso forem conhecidos, então o número médio de vezes que uma determinada tarefa é executada pode ser calculado pelo produto desses dois números. Se o tempo de processamento médio e a correspondente classe de recursos de cada tarefa são conhecidos, então é simples derivar o número total da capacidade por unidade de tempo por papel (assumindo uma utilização de 100%).

4.5.2 Alguma teoria básica de filas para ter em conta a variabilidade

Porque existem sempre flutuações na criação de casos e no tempo de processamento, nem sempre é possível fazer o uso total da capacidade disponível. Não é, por isso, sensato assumir que os recursos são utilizados a 100%. Para ilustrar este aspecto, examinemos um processo que consiste numa só tarefa. Durante cada unidade de tempo, λ novos casos chegam e precisam de ser processados por um recurso. Este recurso é capaz de completar μ casos por unidade de tempo. A capacidade utilizada, ρ, deste recurso é então:

$$\rho = \lambda / \mu$$

Se assumirmos que os tempos de processamento e os tempos de chegada entre casos são distribuídos de uma forma exponencial negativa, o número médio de casos em progresso é *L*, onde:

$$L = \rho/(1-\rho)$$

O tempo de espera médio, *W* – i.e., o tempo de conclusão menos o tempo de processamento – é:

$$W = L/\mu = \rho/(\mu-\lambda)$$

O tempo médio do sistema, *S* – i.e., o tempo de conclusão total (tempo de espera e tempo de processamento) – é:

$$S = W+1/\mu = 1/(\mu-\lambda)$$

Assumindo que uma média de 8 novos casos chegam por hora e que uma média de 10 casos podem ser processados por hora. A capacidade de utilização é então de 80% (ρ = 8/10 = 0.8). Em média, existem 4 casos em progresso (*L* = 4) e o tempo de espera médio é de 24 minutos (*W* = 0.4 horas). Com uma capacidade de utilização de 80%, o tempo de conclusão médio é, assim, de 30 (24 + 6) minutos. Com uma capacidade de utilização de 95% e uma média de tempo de processamento de 6 minutos, o tempo de conclusão médio subiria para não menos do que 2 horas. Este pequeno exemplo mostra que quando a chegada de casos é irregular, não é sensato procurar uma utilização de mais de 80%.

Utilização (ρ)	Número médio em progresso (L)	Utilização (ρ)	Número médio em progresso (L)	Utilização (ρ)	Número médio em progresso (L)
0.10	0.11	0.80	4.00	0.98	49
0.25	0.33	0.85	5.66	0.99	99
0.50	1.00	0.90	9.00	0.999	999
0.75	3.00	0.95	19.00	0.9999	9999

Figura 4.35. O número médio de casos em progresso com base na taxa de utilização

A figura 4.35 mostra o impacto da utilização no número médio de casos em progresso. O impacto resultante da duplicação da utilização de 0.25 para 0.50 (+0.66 casos) é muito menor que o impacto de um pequeno aumento de 0.98 para 0.99 (+50 casos).

A situação anteriormente descrita corresponde a uma *fila M/M/1*. O primeiro *M* indica que os tempos entre chegada de casos são distribuídos de um modo exponencial negativo. O segundo *M* indica que os tempos de processamento também são distribuídos desta forma. O número 1 indica que há apenas um recurso. Para mostrar os quão sensíveis são os tempos de espera à variabilidade dos tempos de processamento, podemos considerar a fila *M/G/1*. Nesta fila, os tempos de processamento são distribuídos aleatoriamente (*G* = geral). Os únicos elementos que sabemos são que o tempo de processamento médio é $1/\mu$ e que o desvio padrão é σ. Baseando-nos nestes dois parâmetros podemos definir o coeficiente de variação, *C*:

$$C = \mu\sigma$$

O coeficiente de variação é uma medida de divergência relativa da média. Quanto maior for *C*, maiores são as diferenças entre tempos de processamento. Na fila *M/G/1* a capacidade de utilização é também igual a $\rho=\lambda/\mu$. Contudo, o número médio de casos em progresso (*L*) depende agora do coeficiente de variação:

$$L = \rho+(\rho^2/(2(1-\rho)))(1+C^2)$$

(Isto é conhecido como a formula de Pollaczek-Khinchin.) O tempo médio de espera, *W*, também depende fortemente do valor de *C*:

$$W = (\rho/(2\mu(1-\rho)))(1+C^2)$$

Estas fórmulas mostram que variações maiores no tempo de processamento podem resultar em maiores tempos de conclusão. Reciprocamente, tempos de processamento mais curtos vão ter tempos de conclusão mais curtos. Para ilustrar este facto, podemos assumir uma situação em que uma média de 8 novos casos chega por hora e o processamento de tempo para cada um é precisamente de 6 minutos. Neste caso, o coeficiente de variação *C* é 0. Pela aplicação das fórmulas descobrimos que a média de tempo de espera é apenas de 12 minutos. Então, os tempos de conclusão dependem fortemente da variação dos tempos de processamento. Note-se que no caso de tempos de processamento distribuídos de acordo com uma exponencial negativa, *C* iguala a 1 e a fórmula de Pollaczek-Khinchin reduz-se à fórmula dada anteriormente.

Fizemos há pouco o uso de várias fórmulas simples da *teoria das filas*, uma teoria que faz parte da disciplina de *Investigação Operacional* (IO). Há muitos resultados da teoria das filas que podem ser aplicados directamente no contexto de gestão de *workflows*. Tal como as filas *M/M/1* e *M/G/1*, discutidas anteriormente, as filas *M/M/n* (as quais contêm vários recursos idênticos) são também fáceis de analisar. Para as filas *M/G/n* e *G/G/n*, existem fórmulas para aproximar o tempo médio de espera. Um resultado que é aplicável a toda a fila

(independentemente da distribuição entre a chegada de casos, a distribuição dos tempos de processamento e o número de recursos) é a *fórmula de Little*:

$$L = \lambda S$$

Esta formula estabelece uma ligação entre o número de casos em progresso, *L*, a chegada entre casos, λ, e o tempo médio do sistema, *S*. Se o tempo de conclusão para um caso é de 5 dias (*S*=5) e uma média de 25 novos casos chegam por dia (λ =25), então o número médio de casos em progresso é de 125 (*L*=125).

Dada uma provisão esperada de casos e várias suposições sobre o seu processamento, podemos usar simulação e/ou a teoria das filas para determinar a capacidade requerida durante um período particular. Baseado nestes requisitos de capacidade pode ser desenhado um plano. Ao prepará-lo, flutuações na criação de casos, perda de recursos temporária e outros problemas, devem também ser tomados em consideração. O mesmo aplica-se ao nível desejado de serviço. Para garantir tempos curtos de conclusão é por vezes necessário aumentar substancialmente o número de recursos.

Existe uma ligação clara entre o planeamento de capacidade num ambiente de *workflow* e num ambiente de produção. Muitos conceitos usados nos sistemas *Manufactoring Resources Planning* (MRR-II) podem ser directamente transferidos para sistemas de gestão de *workflow*. Em vez de uma lista de material necessário (*Bill-Of-Material* – BOM), o ponto de partida é a definição do processo.

EXERCÍCIOS

Exercício 4.1 Optimização do uso de dados

Considere o diagrama do processo sequencial modelado em termos de encaminhamento e papéis na figura 4.36

Existem nove tarefas e os funcionários estão divididos em 3 classes de recurso (papéis): X, Y e Z. Cada tarefa precisa de ser executada por alguém com o papel apropriado.

(a) Modele a definição do processo em termos de uma rede de Petri.
(b) Será o diagrama de encaminhamento/papéis apropriado para a especificação dos processos de *workflow*?

Para a execução do processo de *workflow*, os seguintes 9 elementos de dados são relevantes: D1, D2, …, D9. As relações entre os dados e as tarefas são dadas na matriz CRUD apresentada na figura 4.37.

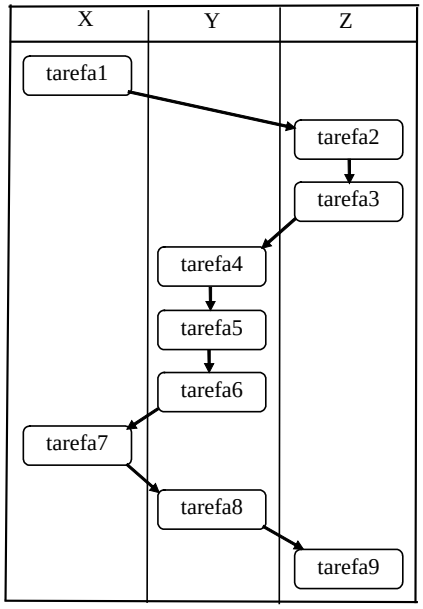


Figura 4.36. Processo

Vamos assumir que só os dados e o seu uso são relevantes para a ordenação das tarefas. O processo sequencial apresentado no diagrama de encaminhamento/papéis está longe de ser óptimo, isto é, a tarefa 4 pode ser executada directamente depois da tarefa 1; não existe necessidade de esperar pela tarefa 2 e 3.

(c) Melhore o processo tornando-o mais paralelo.
(d) É uma boa ideia combinar tarefas? Se sim, que tarefas são candidatas?

	D1	D2	D3	D4	D5	D6	D7	D8	D9
Tarefa1	C	C							
Tarefa2		R				C			
Tarefa3						R	C		
Tarefa4	R		C						
Tarefa5	R	R		C					
Tarefa6		R	R	R	C				
Tarefa7	R	U			R		R		
Tarefa8							R	C	
Tarefa9	R	R						R	C

{C=Create, R=Read, U=Update, D=Delete}

Figura 4.37. Matriz CRUD

Exercício 4.2 Invariantes

Considere as redes de Petri apresentadas nas figuras 4.38, 4.39, 4.40 e 4.41.

(i)

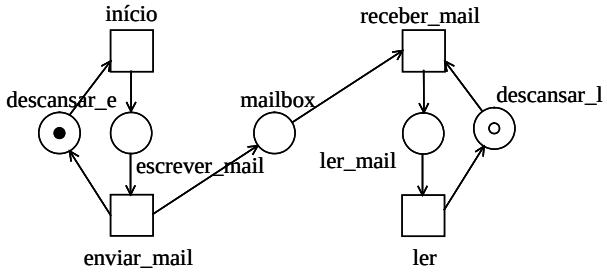


Figura 4.38. E-mail

(ii)

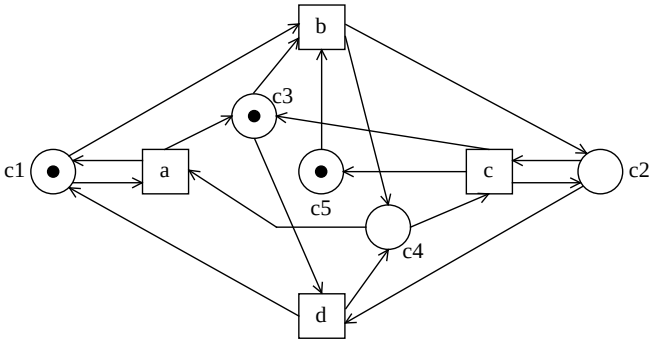


Figura 4.39. Rede

(iii)

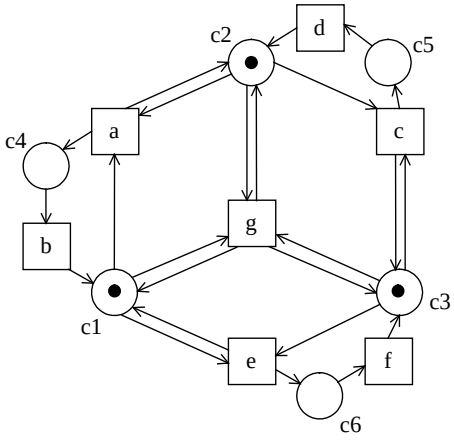


Figura 4.40. Rede

(iv)

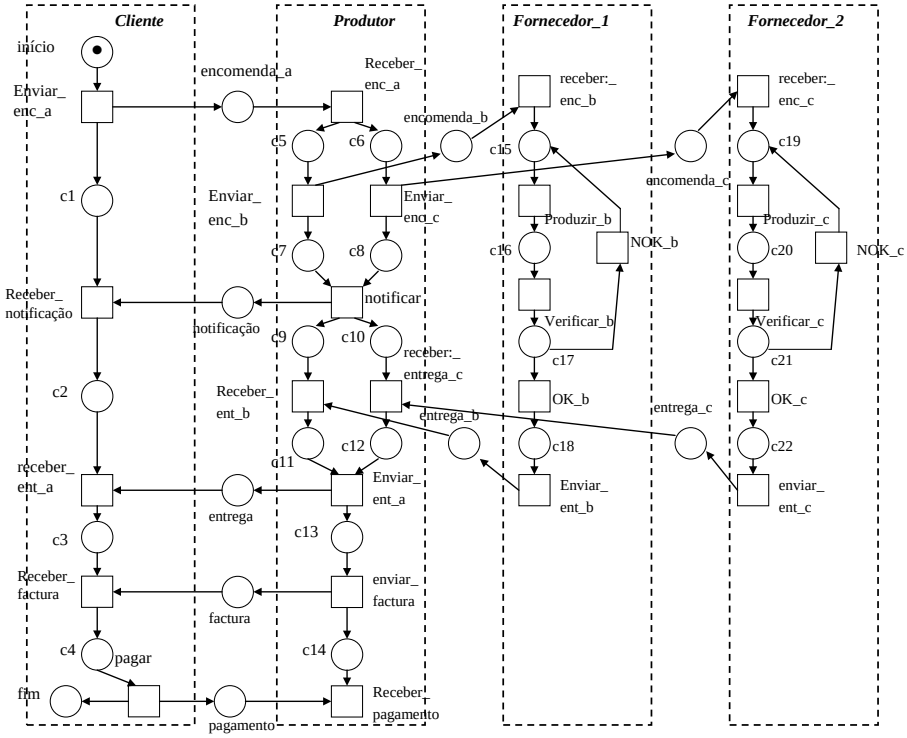


Figura 4.41. Supply chain

Responda para cada rede de Petri às seguintes questões (veja o apêndice A).

(a) Quais é que são os lugares invariantes (máximo 5)? O que mostram?
(b) Quais são as transições invariantes (máximo 5)? O que mostram?
(c) A rede é limitada?
(d) A rede é viva?
(e) A rede é de escolha livre?
(f) Quais são os componentes S?

Exercício 4.3 Verificação da definição de processos

Considere a definição do processo apresentada na figura 4.42.

(a) Confira, construindo o grafo de alcançabilidade, que o processo está correcto.
(b) Estime o número de estados quando é removida a condição *c6*.
(c) Prove através de invariantes de lugar que os dois sub-procedimentos (*t2…t6* e *t7…t12*) não estão activos ao mesmo tempo (exclusão mútua).
(d) Prove que há uma dependência linear entre o *início* e o *fim* (indique as leis de conservação em termos de lugares invariantes).

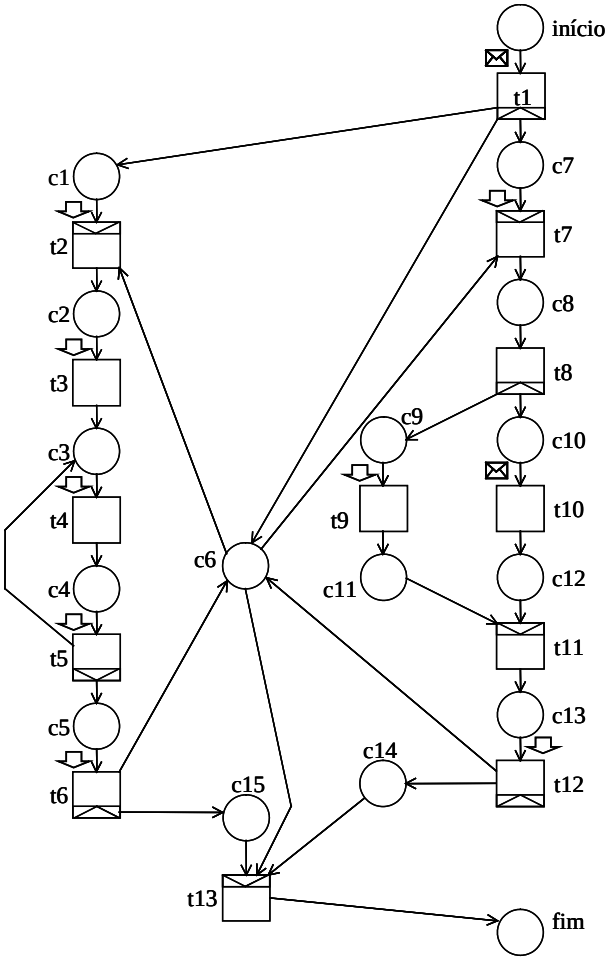


Figura 4.42. Rede

Exercício 4.4 Procura de erros

Considere as definições dos processos apresentadas nas figuras 4.43, 4.44 e 4.45. Responda para cada definição de processo às seguintes questões:

(a) A definição do processo está correcta?
(b) Se não, mostre o erro (grafo de alcançabilidade e/ou lugares invariantes).

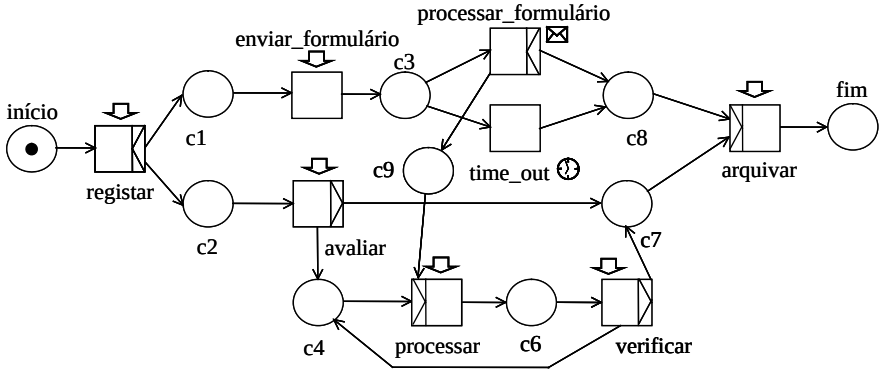


Figura 4.43. Tratamento de reclamações (1)

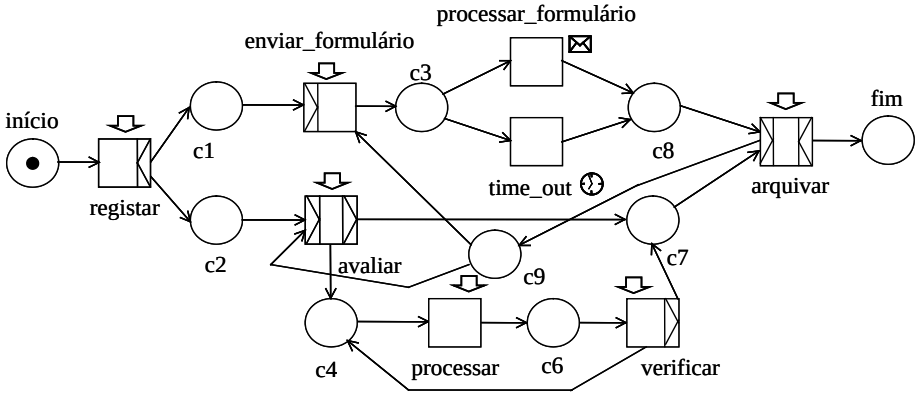


Figura 4.44. Tratamento de reclamações (2)

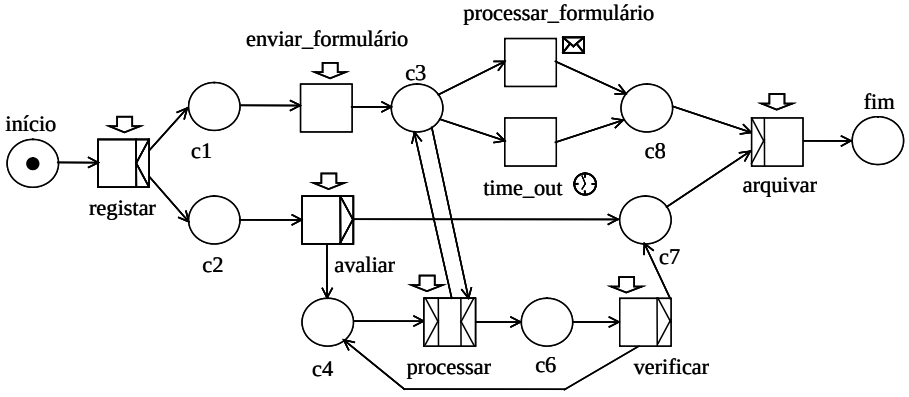


Figura 4.45. Tratamento de reclamações (3)

Exercício 4.5 Análise de desempenho I

Considere o processo da figura 4.46.

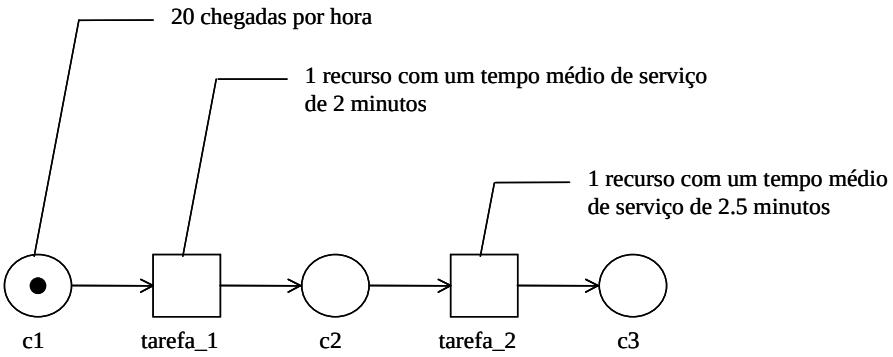


Figura 4.46. Processo (1)

(a) Determine os seguintes indicadores de desempenho:
- Taxa de ocupação (utilização) para cada recurso,
- TEP médio (trabalho em progresso),
- Tempo de fluxo médio (*throughput time*), e
- Tempo de espera médio para cada tarefa.

A tarefa 2 é uma tarefa de verificação. A administração pensa numa execução selectiva desta tarefa onde são conferidos apenas 25% dos casos. O tempo médio de serviço desta nova tarefa é de 6 minutos.

(b) Determine novamente os indicadores de desempenho:
- Taxa de ocupação (utilização) para cada recurso,
- TEP médio (trabalho em progresso),
- Tempo de fluxo médio (*throughput time*), e
- Tempo de espera médio para cada tarefa.

Exercício 4.6 Análise de desempenho II

Considere o processo da figura 4.47.

(a) Determine os seguintes indicadores de desempenho:
- Taxa de ocupação (utilização) para cada recurso,
- TEP médio (trabalho em progresso),
- Tempo de fluxo médio (*throughput time*), e
- Tempo de espera médio para cada tarefa.

Os dois recursos associados a tarefa 1 combinam forças e trabalham juntos tanto em casos fáceis como em casos difíceis. Como resultado, o tempo médio

para executar a tarefa 1 para um caso é de 2 minutos (i.e., uma capacidade total de 4 minutos).

(b) Determine novamente os indicadores de desempenho:
- Taxa de ocupação (utilização) para cada recurso,
- TEP médio (trabalho em progresso),
- Tempo de fluxo médio (*throughput time*), e
- Tempo de espera médio para cada tarefa.

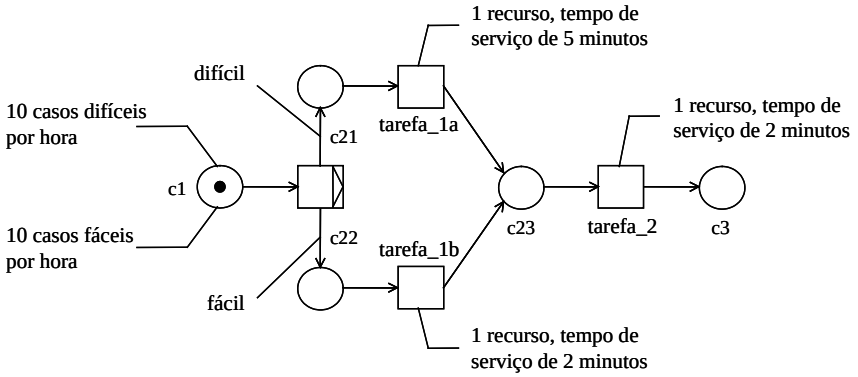


Figura 4.47. Processo (2)

Exercício 4.7 Análise de desempenho III

Considere um processo no qual *ct1* e *ct2* são verificações (ver a figura 4.48). Se forem positivas, a tarefa *bt* (por exemplo, o pagamento de danos) é executada. Se uma delas é negativa, *bt* é ignorada. As duas tarefas de verificação são independentes uma da outra.

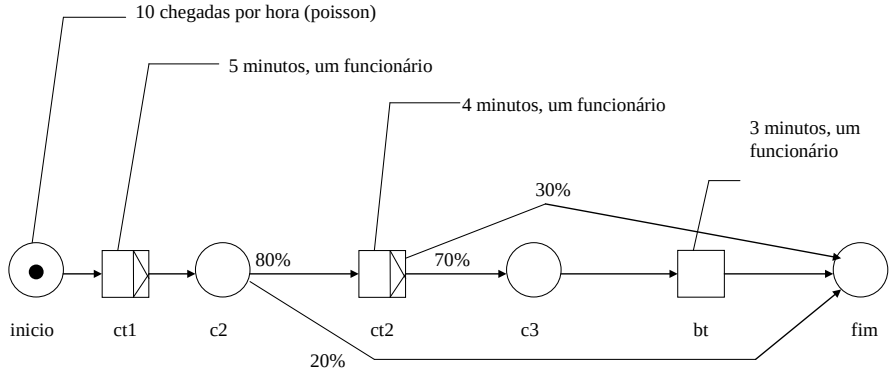


Figura 4.48. Processo (3)

(a) Determine os seguintes indicadores de desempenho:
- Taxa de ocupação (utilização) para cada recurso,
- TEP médio (trabalho em progresso),
- Tempo de fluxo médio (*throughput time*), e
- Tempo de espera médio para cada tarefa.

Dê pelo menos duas alternativas que sejam definições de *workflow* melhoradas.

(b) Para cada alternativa, responda às seguintes questões:
- Porque é a melhor?
- Qual é a utilização dos recursos?
- Qual é o *throughput* máximo?

Exercício 4.8 E-business

No comércio electrónico, *workflows* de diferentes organizações estão juntos. Um deles desempenha o papel de cliente e o outro de servidor. Estes *workflows* são apresentados na figura 4.49.

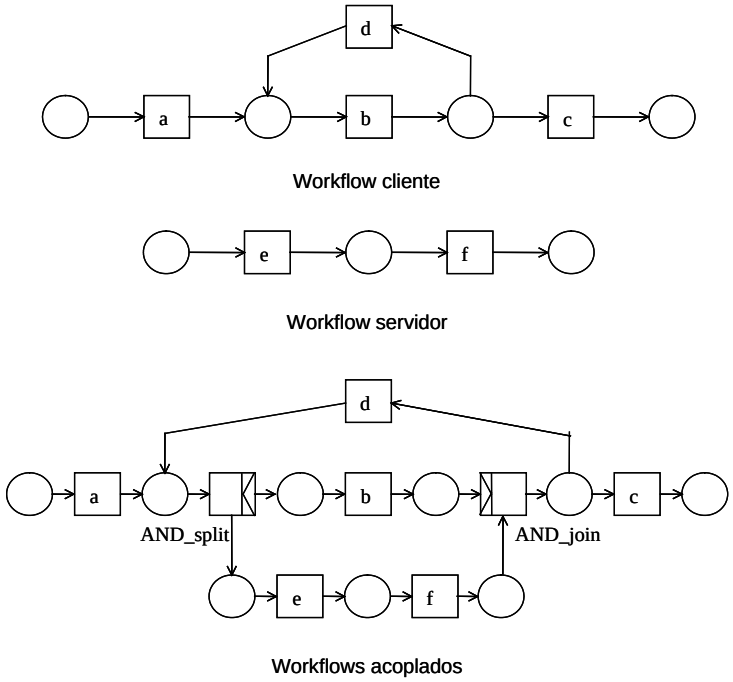


Figura 4.49. Workflows

(a) Apresente derivações para o cliente e para o servidor.

(b) Use estas derivações para obter a derivação conjunta dos workflows (com isto, provamos que esta junção de dois *workflows* é segura e correcta).

Na figura 4.50 vemos novamente a junção entre dois processos: um processo cliente e um processo servidor. Durante o percurso do processo servidor existe uma troca de informação entre o servidor e o cliente: depois da tarefa *d* estar concluída, uma mensagem é enviada de *t* para *q* e depois quando a tarefa *c* é concluída uma mensagem é enviada de *r* para *v*.

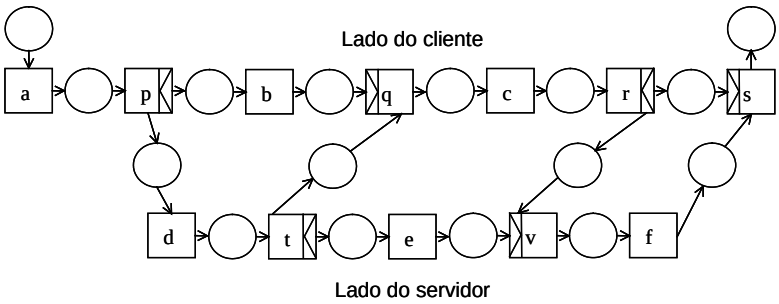


Figura 4.50. Processo (2) cliente/servidor

(a) Existirá uma derivação possível usando os blocos de construção?
(b) O *workflow* é seguro e correcto? Apresente argumentos.

(Página deixada propositadamente em branco)

Capítulo 5.

Funções e Arquitectura dos Sistemas de Workflow

5.1 O Papel dos Sistemas de Gestão de Workflow

Nos capítulos anteriores foi dada uma especial atenção à modelação e aperfeiçoamento dos processos de negócio. Foram apresentadas técnicas para os descrever de uma forma estruturada, para os analisar e para os aperfeiçoar. Claramente, estas técnicas são a chave para atingir melhorias drásticas na eficiência e na eficácia da organização e no seu desempenho. Uma questão óbvia que se põe é como realizar um desejado processo de negócio utilizando tecnologias de informação. Ao utilizar esta abordagem é importante não se perder de vista os benefícios de uma abordagem orientada ao processo. O sistema de informação deve ser estruturado de tal forma que possa lidar com possíveis mudanças no futuro. Na prática, isto significa que os sistemas de informação têm de obedecer aos seguintes requisitos:

- Os sistemas de informação têm de ser implementados de maneira que a estrutura dos processos de negócio esteja claramente reflectida. Isto torna o processo verificável pelo utilizador e reduz as hipóteses de ocorrerem erros, tanto durante a fase de desenvolvimento do sistema, como durante a fase de execução do processo.
- Deve haver uma abordagem integrada que engloba também tarefas não realizadas por computadores. Os processos de negócio na actualidade expandem frequentemente muito para além do que era tradicionalmente guardado num sistema de informação.
- Os sistemas de informação têm de ser implementados de tal forma que a estrutura do processo de negócio possa ser facilmente modificada. Isto permite que as organizações respondam com flexibilidade ao seu ambiente em constante mutação e reestruturem os seus processos de negócio.
- É importante que o desempenho de um processo de negócio possa ser acompanhado para que quaisquer problemas possam ser reconhecidos cedo no processo. As intervenções devem ser facilmente efectuadas e possíveis no momento em que algo corre mal. Nesta perspectiva, o desempenho de um processo de negócio deve ser fácil de medir e possível de melhorar.
- A atribuição de trabalho às pessoas é um ponto de particular interesse. Uma boa gestão da carga de trabalho é importante para se conseguirem processos de negócio eficientes e eficazes.

5.1.1 De que forma os sistemas de informação são tradicionalmente estruturados

Tradicionalmente a gestão dos processos não tem sido separada das aplicações dos sistemas de informação. Por outras palavras, a gestão dos processos tem sido escondida dentro dos próprios sistemas de informação. Dado que pouca atenção tem sido dada à estrutura dos processos dentro dos sistemas tradicionais, frequentemente tem sido difícil de identificar efectivamente os processos de negócio. Ainda para mais, os processos contidos nos sistemas estão frequentemente incorrectos ou incompletos.

5.1.2 Separação da gestão e da execução

Um passo importante no sentido de conseguir sistemas de informação que possam cumprir os requisitos descritos anteriormente é proceder à sua divisão num subsistema que trata da *gestão* dos processos de negócio ("sistema de gestão" ou "sistema logístico") e outro subsistema que trata da *execução* das tarefas de um processo de negócio específico (a "aplicação"; ver figura 5.1). O sistema de gestão trata da conclusão lógica dos casos, sem proceder efectivamente à execução de tarefas. Assegura que nenhum passo é ignorado, que todos os passos são executados pela ordem correcta, que as tarefas são executadas em paralelo quando possível, que as aplicações correctas são utilizadas para suportar cada tarefa e assim sucessivamente. O sistema de gestão também assegura que o trabalho é distribuído pelos funcionários, considera as respectivas ausências, suporta a separação de funções e nível de autorização e assim sucessivamente.

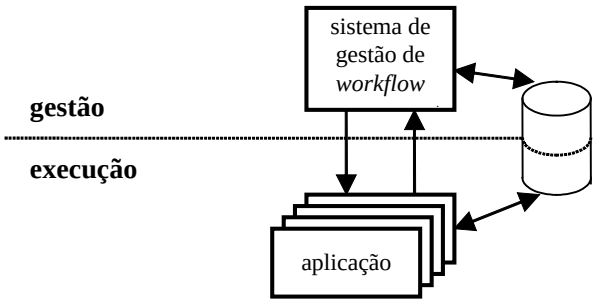


Figura 5.1. A separação entre a logística e a execução

À parte da estrutura do processo de negócio, o sistema de gestão não tem na realidade quaisquer características específicas de aplicação. Para *diferenciar* gestão de execução, neste livro utilizamos o princípio de que a gestão apenas pode *consultar* os atributos do caso para *tomar decisões de encaminhamento*. Consideramos modificar os atributos do caso como parte da execução em detrimento da gestão.

Atribuir o trabalho (i.e., os itens de trabalho) às pessoas ou aplicações apropriadas no momento correcto é uma função do sistema de gestão, para que

as tarefas de um caso específico possam ser executadas. O sistema de gestão logístico interage com o utilizador, reage aos estímulos do seu ambiente (por exemplo, a chegada de uma mensagem EDI), ou executa tarefas automáticas ou dependentes do tempo (e em princípio, uma tarefa dependente do tempo também espera por um estímulo do ambiente). Quando uma aplicação de suporte a um passo específico do processo é definida, o sistema de gestão começa a sua execução de uma forma correcta. Uma aplicação suporta o utilizador na execução da tarefa. O sistema de gestão e as aplicações comunicam utilizando os atributos do caso. Quando uma aplicação é executada, os atributos podem ser analisados, alterados e transferidos às tarefas seguintes. Quando a aplicação conclui a sua execução, qualquer alteração nos atributos do caso é transferida de volta para o nível de gestão.

5.1.3 Vantagens

A separação entre a gestão e as aplicações tem um número importante de vantagens:

- Possibilita atingir uma gestão uniforme da funcionalidade, isolando a gestão do resto do sistema (tradicionalmente, esta funcionalidade estava distribuída no sistema de informação). Isto torna possível a reutilização da mesma funcionalidade em diversas tarefas.
- As aplicações deixam de necessitar de funcionalidades de gestão, tornando-se mais simples e completamente independentes do seu contexto. Isto torna possível modificar o processo de negócio numa fase mais tardia.
- A camada de gestão torna possível a integração de aplicações variadas. Desta maneira é até possível integrar novas aplicações com sistemas de gerações anteriores (sistemas legados).
- Ao nível da gestão, o processo de negócio é identificável e o estado de um caso particular é fácil de estabelecer. Logo, torna-se mais fácil acompanhar o processo. Dado que é mais simples verificar ao nível da gestão quais as tarefas que têm de ser executadas, é mais fácil determinar quem deveria fazer o quê para cada caso particular. A execução do processo é mais flexível, tornando mais fácil de verificar o seu progresso e os seus pontos críticos (e.g., *bottlenecks*).

5.1.4 Software de gestão de workflow

Dado que o processo de gestão das funcionalidades deve, em princípio, ser bastante genérico, ao invés de específico, para uma aplicação, torna-se atractivo utilizar *software* genérico: sistemas de gestão de *workflow*. Estes podem interpretar e aplicar a estrutura do processo e as regras de distribuição da carga de trabalho.

Há um grande número de sistemas de gestão de *workflow standards* no mercado. Estes diferenciam-se largamente pelas funcionalidades que oferecem. Neste capítulo, tentaremos indicar – em termos gerais – as funcionalidades que devem ser esperadas de sistemas de gestão de *workflow*. Adicionalmente, vamos

analisar os aspectos técnicos que são importantes na selecção e introdução de um sistema deste tipo.

5.2 Um Modelo de Referência

Tal como vimos no capítulo 1, os sistemas de gestão de *workflow* permitem a "extracção" da gestão dos processos das aplicações de *software*. Até certo ponto, é possível comparar tais sistemas com um sistema gestor de base de dados. Afinal, os sistemas gestores de bases de dados tornam possível a extracção da gestão de dados da aplicação de *software*. Ambos os tipos de sistemas suportam uma quantidade de funcionalidades genéricas. Porque – ao contrário dos sistemas gestores de bases de dados – os sistemas de gestão de *workflow* emergiram há pouco tempo, em vários aspectos é ainda incerto quais os componentes que fazem parte das funcionalidades básicas do sistema. A tecnologia é ainda nova e não está totalmente estabelecida.

Ainda para mais, a gestão de *workflow* possui muitas "faces". Os sistemas de gestão de *workflow* podem ser implementados de forma a atingir flexibilidade, integração de sistemas, optimização de processos, mudanças na organização, manutenção melhorada, desenvolvimento evolucionário e assim sucessivamente. Tudo isto significa que pode facilmente surgir confusão quanto ao que pode ser esperado da funcionalidade de um sistema de gestão de *workflow*. Este perigo foi reconhecido numa fase inicial pelo *Workflow Management Coalition* (WFMC) – uma organização cujo papel inclui a normalização da terminologia de gestão de *workflows* e a definição de normas para a troca de dados entre os sistemas de gestão de *workflow* e aplicações. Em 1996, o WFMC tinha já duzentos membros (incluindo muitos fornecedores de produtos de gestão de *workflow*).

Um dos muitos princípios utilizados pelo WFMC é o *modelo de referência de workflow*. Este modelo é uma descrição geral da arquitectura de um sistema de gestão de *workflow* em que os vários componentes e as interfaces são identificadas e resumidas. A figura 5.2 ilustra o modelo de referência para sistemas de *workflow*.

O modelo mostra que a parte central de um sistema de *workflow* é o *serviço de execução* de *workflows*. Esta parte do sistema bombeia os casos através da organização. O serviço de execução assegura que as actividades correctas são executadas pela ordem estabelecida e pelas pessoas certas. De forma a atingir este objectivo é feito uso das definições de processos e das classificações dos recursos produzidos pelas *ferramentas de definição de processos*. Além de ilustrar os processos e a organização, estas ferramentas frequentemente oferecem outras facilidades e técnicas de análise tais como a simulação. Os itens de trabalho são fornecidos aos funcionários através das *aplicações de workflow cliente*. Ao seleccionar um item de trabalho, um funcionário pode começar a executar uma determinada tarefa para um caso específico. Quando executar uma tarefa pode ser necessário iniciar uma aplicação. Todas as aplicações de *software* que podem ser iniciadas a partir do sistema de *workflow* são conhecidas por aplicações invocadas no modelo de referência. A verificação de *workflows*, o controlo de casos e a gestão dos funcionários são suportadas pelas *ferramentas de administração e monitorização*.

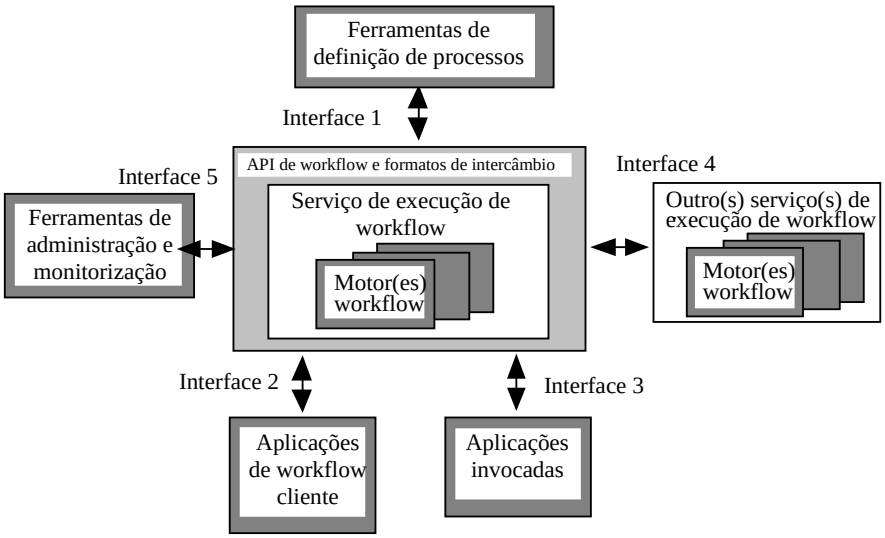


Figura 5.2. O modelo de referência do Workflow Management Coalition (© WFMC)

Cinco interfaces são também ilustradas na figura 5.2. O WFMC tem-se empenhado em normalizar estas interfaces. Ao criar um sistema de informação baseado num sistema de gestão de *workflow*, as *Interfaces 3* e *4* são de particular interesse. A primeira está associada ao controlo de aplicações pelo sistema de *workflow*; a segunda está associada ao intercâmbio de casos (ou partes de casos) entre sistemas de *workflow* autónomos. As outras interfaces são maioritariamente utilizadas pelo próprio sistema de gestão de *workflow*.

A figura 5.2 ilustra apenas uma pequena perspectiva das funcionalidades de um sistema de gestão de *workflow*. Iremos clarificar e explorar cada um dos componentes.

5.2.1 Serviço de execução de workflow

O serviço de execução de *workflow* é o centro do sistema de *workflow*. Este componente cria novos casos, gera itens de trabalho baseados na descrição do processo, associa recursos com itens de trabalho, suporta o desempenho das actividades e permite o registo de aspectos particulares do *workflow*. Por razões técnicas, o serviço de execução pode englobar vários motores de *workflow*. A sua utilização pode, por exemplo, melhorar a expansão de todo o sistema. Num serviço de execução com mais que um motor de *workflow*, o trabalho é distribuído entre os vários motores. Esta distribuição pode ser baseada nas características do caso, da tarefa e/ou do recurso. Em geral, o utilizador não irá notar quando um sistema de *workflow* está a usar mais do que um motor.

Motor de *workflow* Um motor de *workflow* fornece as facilidades necessárias para a conclusão logística dos casos. Em determinados casos, vários motores de *workflow* funcionam lado a lado. Cada um lida com uma parte dos

casos e/ou processos. As responsabilidades de um motor de *workflow* englobam:

- Criar novos casos e remover os casos concluídos;
- Encaminhar casos com base na interpretação da definição dos processos;
- Gerir atributos de casos;
- Submeter itens de trabalho aos recursos correctos (funcionários), com base na classificação dos recursos;
- Gerir e tratar os disparos;
- Iniciar aplicações de software durante a execução de uma actividade;
- Registar dados históricos;
- Fornecer um resumo do *workflow*; e
- Monitorizar a consistência do *workflow*.

Os motores de *workflow* são assim o "núcleo" do sistema de *workflow*, sem o qual este não poderia operar.

5.2.2 Ferramentas de definição de processos

Um motor de *workflow* tem por base uma ou mais definições de *workflows*. Nos capítulos 2 e 3, vimos que a definição de um *workflow* está dividida em duas partes importantes: a definição do processo (capítulo 2) e a classificação dos recursos (capítulo 3). No modelo de referência de *workflow*, as ferramentas para construir estas definições e classificações são conhecidas por *ferramentas de definição de processos*. Para além de ferramentas para ilustrar *workflows*, é também frequentemente possível fazer uso de ferramentas de análise. No capítulo 4, mostramos que técnicas de análise são aplicáveis no contexto da gestão de *workflow*. Em princípio, podemos diferenciar entre três tipos de ferramentas: (1) *ferramentas de definição de processos*, (2) *ferramentas de classificação de recursos*, e (3) *ferramentas de análise*. Nalguns sistemas de gestão de *workflow*, estas três ferramentas estão integradas numa única aplicação de definição de *workflow* e análise. Repare-se que o termo "ferramentas de definição de processos" utilizado pelo WFMC é algo confuso, dado que ele abrange não só as ferramentas para a modelação de definições de processo, mas também ferramentas de classificação de recursos e ferramentas de análise.

A ferramenta de definição de processos Um processo é especificado utilizando uma ferramenta de definição de processos. O capítulo 2 examinou processos definidos em termos de redes de Petri. Em muitos sistemas de gestão de *workflow*, no entanto, os processos são formulados de uma forma diferente. Contudo, na maior parte dos casos é fácil relacionar as construções de encaminhamento utilizadas em redes de Petri. O poder expressivo destes métodos alternativos de modelação é tipicamente mais fraco, porque certas estruturas de encaminhamento são excluídas. Por exemplo, muitos sistemas de gestão de *workflow* abstraem-se do modelo explícito de estados e isto não permite que formas de encaminhamento tais como o OR-split implícito sejam modeladas. As funcionalidades básicas das ferramentas de definição de processos consistem nos seguintes elementos:

- Capacidade para construir definições de processos (nome, descrição, data da versão, componentes, e assim sucessivamente);
- Capacidade em modelar o encaminhamento sequencial, paralelo, selectivo e iterativo através de componentes gráficos tais como o AND-split, o AND-join, o OR-split e o OR-join;
- Suporte à gestão de versões (afinal, podem existir várias versões do mesmo processo);
- Definição de atributos de casos utilizados no processo;
- Especificação de tarefas; e
- Verificar se um processo está definido de forma correcta (sintacticamente) e se existem omissões ou inconsistências.

É ncessário estabelecer um conjunto de características para cada tarefa num processo. Este conjunto determina as condições em que uma tarefa pode ser executada e que operações podem ser efectuadas. Para cada tarefa está estabelecido o seguinte:

- O nome e a descrição da tarefa;
- A informação sobre a tarefa – por outras palavras, qualquer instrução e informação que apoiem os funcionários na realização de tarefas;
- Os requisitos dos recursos que executam a tarefa (por exemplo, uma especificação do seu papel, a unidade organizacional, ou informação sobre a separação de funções);
- As características de encaminhamento da tarefa (AND-split, AND-join, OR-split, OR-join);
- A especificação de quaisquer disparos necessários;
- As instruções para o motor de *workflow* (por exemplo, prioridades, gestão de casos e gestão de recursos);
- As aplicações que podem ser executadas, mais as condições e a ordem em que elas devem ser realizadas;
- A especificação dos atributos dos casos que são usados e ajustados pela aplicação; e
- As regras de decisão que determinam as tarefas subsequentes com base nos atributos do caso quando existe um OR-split ou uma mistura OR/AND-split.

O processo estabelecido utilizando a ferramenta de definição de processo é o ponto mais importante do *workflow*.

A ferramenta de classificação de recursos Além de definir o processo, os recursos necessários para a execução de um *workflow* têm de ser classificados para que as tarefas possam ser separadas dos funcionários. A maior parte dos sistemas de gestão de *workflow* fornece uma ferramenta de classificação de recursos em que a relação entre as várias classes de recursos pode ser representada graficamente. Desta forma, os seguintes objectos podem ser estabelecidos:

- Uma lista das classes de recursos, frequentemente subdividida em papéis (baseados em qualificações, funções e aptidões) e unidades organizacionais (baseadas na distribuição em equipas, sucursais e/ou departamentos);
- As características específicas de uma classe de recursos; e
- O relacionamento entre as várias classes de recursos (por exemplo, uma hierarquia de papéis ou unidades organizacionais).

A ferramenta de análise Antes que um *workflow* possa "entrar em produção" é útil analisá-lo primeiro. Essa análise pode englobar a verificação da correcção semântica da definição do processo e também uma simulação de modo a obter uma melhor perspectiva sobre o tempo de conclusão dos casos. Em geral, a geração actual de sistemas de gestão de *workflow* oferece possibilidades de análise limitadas. Na maior parte dos sistemas é, portanto, possível definir *workflows* que poderiam ter consequências graves se fossem de facto postos em prática. Contudo, tal como foi descrito no capítulo 4, é possível aplicar técnicas avançadas de análise. Futuros sistemas de gestão de *workflow* irão oferecer mais e melhores possibilidades de análise.

5.2.3 Aplicações de workflow cliente

Os funcionários que estão apenas envolvidos na actual execução de um processo nunca irão utilizar as ferramentas de definição de processos. O único contacto que têm com o sistema de *workflow* é através das *aplicações de workflow cliente*. Cada funcionário possui uma *lista de trabalho* (também conhecida por "in-tray" ou "in-basket") que faz parte das aplicações de *workflow* cliente. O motor de *workflow* utiliza esta lista de trabalho para mostrar que itens de trabalho precisam de ser realizados. Ao seleccionar um item de trabalho, um funcionário pode iniciar a execução de uma tarefa para um caso específico. Em princípio, cada funcionário tem uma lista de trabalho pessoal que indica todo o trabalho que ele ou o seu grupo tem de realizar. Por esse motivo, a lista de trabalho forma o elo mais forte entre o trabalho e o funcionário.

Tal como foi mostrado no capítulo 3, a distribuição de trabalho pode ser orientada a um modelo *push* ou *pull*. Verifica-se o primeiro modelo quando o motor de *workflow* distribui itens de trabalho aos funcionários individualmente. O segundo modelo dá-se quando os itens de trabalho são distribuídos por grupos de funcionários. Desta forma um item de trabalho pode vir a aparecer em várias listas de trabalho. As funcionalidades básicas que devem ser fornecidas por uma lista de trabalho estão enumeradas a seguir:

- Apresentação dos itens de trabalho que podem ser executados por um funcionário;
- Fornecer as propriedades relevantes de um item de trabalho, tais como informações sobre o caso e a tarefa;
- Aptidão para sortear e seleccionar itens, com base nestas propriedades;
- Fornecer informação sobre o estado relativo do motor de *workflow*;
- Começar uma tarefa para um caso específico quando um item de trabalho é seleccionado; e

- Capacidade de comunicar a conclusão de uma actividade (i.e., um determinado item de trabalho).

Em adição, a lista de trabalho pode permitir congelar ou passar um item de trabalho. Tem também de ter a capacidade de lidar com falhas no sistema. A figura 5.3 ilustra uma lista de trabalho do sistema de gestão de *workflow* COSA.

A maior parte dos sistemas de gestão de *workflow* oferecem a *lista de trabalho standard*. Em alguns casos, contudo, é necessário criar uma lista de trabalho personalizada para ambientes específicos.

A lista de trabalho standard A lista de trabalho *standard* oferece as funcionalidades que acabamos de descrever. Dado que não está personalizada para uma situação de negócio específica, as funções disponíveis são genéricas. Em muitas situações, contudo, é possível utilizar parâmetros para a lista de trabalho *standard*. Possibilita, por exemplo, que se possa modificar a disposição e o conteúdo da janela. Algumas listas de trabalho *standard* permitem representar graficamente o estado (logístico) de um caso.

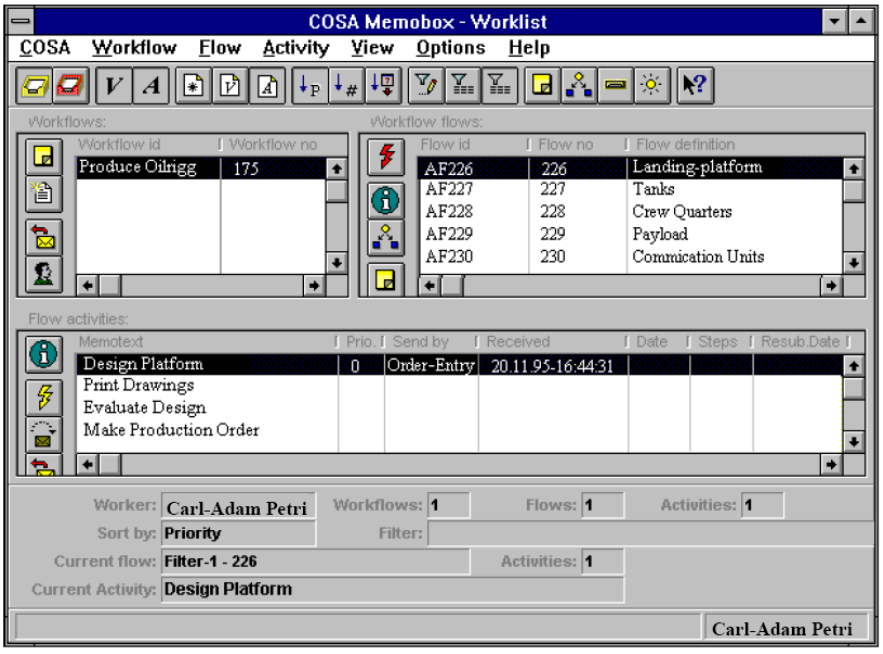


Figura 5.3. Um exemplo de uma lista de trabalho (COSA, © Software-Ley)

A lista de trabalho integrada A única forma de um utilizador típico aceder ao sistema de *workflow* é através da sua lista de trabalho. Quando um sistema está a suportar o trabalho de, digamos, cem funcionários, a apresentação deste componente merece uma especial atenção. Isto pode justificar o desenvolvimento de uma lista de trabalho personalizada adaptada à situação específica do negócio em detrimento da lista de trabalho *standard*. Esta lista de

trabalho personalizada teria de englobar facilidades de suporte em adição às funcionalidades *standard* descritas anteriormente. É por isso que é referida como uma lista de trabalho integrada. Esta lista pode, por exemplo, fazer uso de dados auxiliares para fornecer um suporte extra. Considerações ao nível da segurança e qualidade podem também catalisar o desenvolvimento de uma *lista de trabalho integrada*. O mesmo aplica-se à necessidade de processamento de itens de trabalho em lotes ou em cadeia.

O *processamento em lotes* acontece quando um funcionário é capaz de executar um número de itens de trabalho do mesmo tipo (por outras palavras, repetir a mesma tarefa) sem ter de voltar à lista de trabalho. Isto permite a execução de uma determinada tarefa rotineira várias vezes sucessivamente. O processamento em cadeia acontece quando um funcionário é capaz de executar um número de tarefas sucessivas para um caso específico. Desta forma, não é necessária uma habituação contínua a um novo caso. O *processamento em lotes e em cadeia* evitam uma mudança desnecessária entre a lista de trabalho e a aplicação. Este procedimento pode conduzir a ganhos em termos de eficiência.

5.2.4 Aplicações invocadas

O desempenho de uma tarefa pode dar origem à execução de uma ou mais aplicações. Estas não fazem parte do sistema de gestão de *workflow* porque estão associadas com o real desempenho do trabalho e não com a sua gestão logística. Mas, tais aplicações pertencem ao sistema de *workflow*, que engloba as aplicações, os ficheiros de configuração, o sistema de gestão de *workflow*, as bases de dados, e assim sucessivamente. As aplicações são colocadas em execução pelo motor de *workflow* de maneira a realizar uma determinada tarefa. Como tal, a informação que diz respeito ao caso pode ser submetida. A aplicação pode, por exemplo, fazer uso de um particular valor de um atributo do caso. A identificação do caso é frequentemente usada para procurar a informação apropriada numa base de dados. Adicionalmente, as aplicações também podem mudar os valores dos atributos do caso. Estes atributos modificados são frequentemente utilizados para determinar o encaminhamento do caso. Em geral, é feita uma distinção clara entre as aplicações *interactivas* e as aplicações totalmente *automáticas*.

Aplicação interactiva Uma aplicação interactiva é sempre iniciada em resultado da selecção de um item de trabalho da lista de trabalho. Pode ser uma ferramenta banal de escritório tal como um processador de texto ou uma folha de cálculo, ou um programa desenvolvido especificamente para o processo de negócio (por exemplo, um questionário electrónico que precisa de ser preenchido).

Aplicação totalmente automática Uma aplicação totalmente automática não precisa de interagir com o utilizador. Pode, por isso, ser parte de uma tarefa que pode ser executada sem qualquer intervenção do utilizador. Um exemplo pode ser um programa que executa cálculos complicados (tais como determinar a quantia de um pagamento).

5.2.5 Outros serviços de execução de workflow

Um sistema de *workflow* pode conter vários motores de *workflow*. Estes fazem parte da mesma gestão e utilizam as mesmas definições de *workflow*. Tais motores são designados como fazendo parte do mesmo domínio de *workflow*. Contudo, é também possível ligar vários sistemas de *workflow* autónomos entre si. Desta forma, casos (ou partes de casos) podem ser transferidos de um sistema para o outro. Isto significa que os serviços de execução de *workflow* de cada sistema estão ligados entre si. Normalmente designamos este processo de *interoperabilidade a nível dos workflows*. No futuro, espera-se que mais sistemas de *workflow* estejam interligados. Estes poderão estar em diferentes sucursais da mesma empresa ou em empresas diferentes.

5.2.6 Ferramentas de administração e de monitorização

O serviço de execução de *workflow* assegura o processamento de casos com base em definições de *workflow*. A supervisão e a gestão operacional destes fluxos (incluindo os recursos) são feitas utilizando *ferramentas de administração e monitorização*. Estas podem ser divididas em dois conjuntos: aquelas utilizadas para a gestão operacional dos *workflows* e aquelas utilizadas para o registo e a criação de relatórios. Em muitos sistemas de gestão de *workflow* estas ferramentas estão integradas numa única aplicação.

A ferramenta de gestão operacional A gestão operacional engloba todas as operações pertencentes à gestão dos *workflows*. Logo, não é possível utilizar a *ferramenta de gestão operacional* para modificar a estrutura de um processo de negócio. Podemos subdividir a informação associada a gestão operacional em informação relacionada com os casos e aquela que não o é (i.e. relacionada com recursos ou com o sistema). As funções da ferramenta de gestão operacional relacionadas com a informação de recursos incluem:

- Adição ou remoção de funcionários; e
- Entrada/revisão dos detalhes de um funcionário (nome, endereço, número de telefone, papel, unidade organizacional, autorização e disponibilidade).

Outras funções da ferramenta de gestão operacional são:

- Implementação de novas definições de *workflow*; e
- Reconfiguração do sistema de *workflow* (definição de parâmetros técnicos do sistema).

Note-se que consideramos os detalhes individuais de um funcionário como sendo parte da gestão operacional. O ajustamento da informação que diz respeito à disponibilidade do funcionário como resultado da revisão do horário, das férias ou da ausência por doença são exemplos de gestão operacional relacionada com os recursos. São também necessárias funções para efectuar a gestão operacional dos casos:

- Inspecção do estado logístico de um caso; e
- Manipulação do estado logístico de um caso por razões de força maior, tais como problemas e circunstâncias especiais.

A ferramenta de gestão operacional é, por conseguinte, também utilizada para proporcionar soluções *ad hoc* para problemas originados por falhas do sistema e congestionamentos no processo.

A ferramenta de registo e criação de relatórios Muitos aspectos podem ser registados e guardados durante a execução de um *workflow*. Estes compreendem dados históricos que podem ser úteis à gestão. Por exemplo, os seguintes *indicadores de desempenho* podem ser deduzidos a partir dos dados:

- Média do tempo de execução de um caso;
- Média do tempo de espera e processamento (possivelmente subdivididos por tarefa);
- Percentagem dos casos finalizados dentro de um período predefinido; e
- Média de utilização da capacidade dos recursos.

Note-se que em muitas situações não só as médias mas também as variâncias destes indicadores de desempenho são de importância primordial.

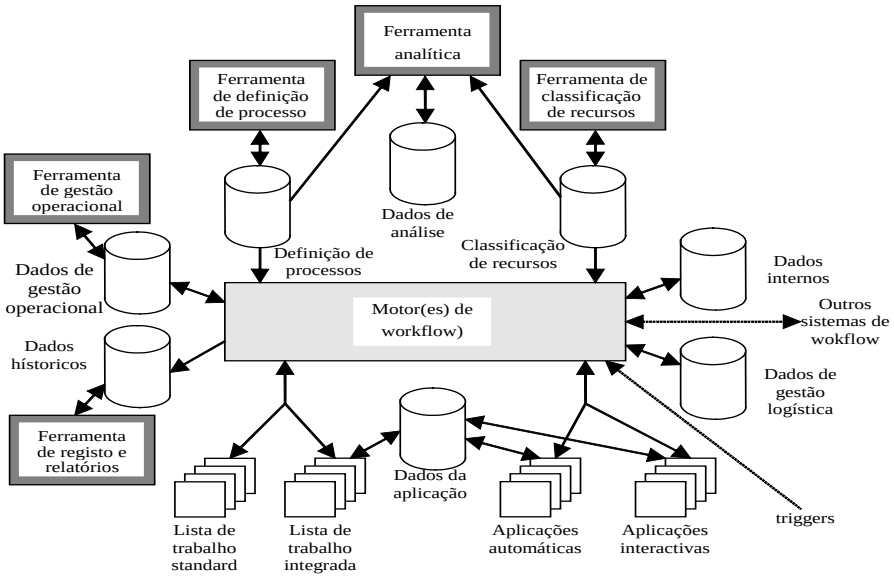


Figura 5.4. Os vários componentes de um sistema de *workflow*

A informação sobre as propriedades dos *workflows* completados é de importância vital para a gestão. Avisos sobre congestionamentos e excesso de capacidade podem dar origem à revisão do processo. Os dados em bruto são fornecidos pelo serviço de execução de *workflow*, e seguidamente, são tratados

pela *ferramenta de registo e criação de relatórios*. Essa pode, por exemplo, decidir a informação que deve ser guardada, assim como, frequentemente, disponibilizar opções para a criação de relatórios. Alguns sistemas de gestão de *workflow* utilizam relatórios predefinidos que são produzidos em intervalos de tempos regulares. Outros oferecem um gerador integrado de relatórios. Isto permite ao utilizador definir relatórios com base na informação guardada. Outros sistemas não fornecem deliberadamente opções para gerar relatórios. Desta forma os dados guardados podem ser acedidos através de um vulgar sistema gestor de base de dados ou um gerador de relatórios genérico. Frequentemente, uma grande quantidade de dados necessita de ser traduzida de forma a produzir informação relevante para a gestão. Claramente existe uma ligação com o *data mining,* o *data warehousing,* e o *OLAP* (*on-line analytical processing*).

A figura 5.4 ilustra a relação entre as ferramentas descritas, sendo esta uma versão mais detalhada do modelo de referência de *workflow* apresentado na figura 5.2. A figura, não indica, contudo, que a ferramenta de análise e a ferramenta de registo e de criação de relatórios fazem frequentemente uso da informação uma da outra. Por exemplo, os dados históricos podem ser utilizados quando analisamos um *workflow* (através, por exemplo, de uma simulação). Os resultados analíticos podem também ser utilizados em pesquisas dedicadas à procura de informação útil à gestão.

5.2.7 Papéis das pessoas envolvidas

A figura 5.4 claramente mostra que um sistema de *workflow* é constituído por muitos componentes que são utilizados por um grande leque de pessoas. Em teoria, existem quatro tipos de utilizadores:

- *O designer do workflow*. O designer do *workflow* usa as ferramentas de definição de processos (por outras palavras, a ferramenta de definição de processo, a ferramenta de classificação de recursos e as ferramentas de análise). Este designer trabalha na estrutura de um *workflow*.
- *O administrador*. O administrador utiliza a ferramenta de gestão operacional. Entre as suas actividades típicas incluem-se a adição de funcionários, a atribuição e eliminação de autorizações, a implementação de novos processos, a monitorização de *workflows* e a resolução de problemas de congestionamentos.
- *O analista de processo*. O analista de processo utiliza a ferramenta de registo e criação de relatórios para informar a gestão sobre o desempenho dos *workflows*. Ao agregar dados detalhados em indicadores de desempenho é possível tornar mais claro o processo de negócio que é suportado pelo sistema de gestão de *workflow*.
- *O funcionário*. A execução do trabalho é realizada pelos funcionários. Neste livro, eles também são referidos por recursos (humanos). Tais recursos são meios de produção escassos que necessitam de ser utilizados da melhor forma possível.

Tal como os quatro tipos de utilizadores, outras pessoas são frequentemente envolvidas na estruturação, gestão e desempenho dos *workflows*. Os utilizadores

do sistema de gestão de *workflow* são geralmente administrados por um gestor. *Workflows* novos e/ou revistos frequentemente requerem aplicações novas ou melhoradas. Os requisitos de informação podem ser também alterados pela introdução de um novo processo. É por esta razão que designers/programadores de bases de dados e designers/programadores de aplicações são também envolvidos na (re)estruturação de um *workflow*. A figura 5.5 ilustra os diversos tipos de pessoas envolvidas no design, implementação e execução de um *workflow*

Como nota, diga-se que, na prática, a distinção entre pessoas e papéis não é sempre precisa como a figura 5.5 deixa transparecer. O analista de processos pode também ser um gestor, um funcionário ou também um administrador – e podem existir vários tipos de administradores. No capítulo 6 iremos examinar com maior detalhe os vários tipos de pessoas envolvidas na implementação e na gestão dos sistemas de gestão de *workflow*.

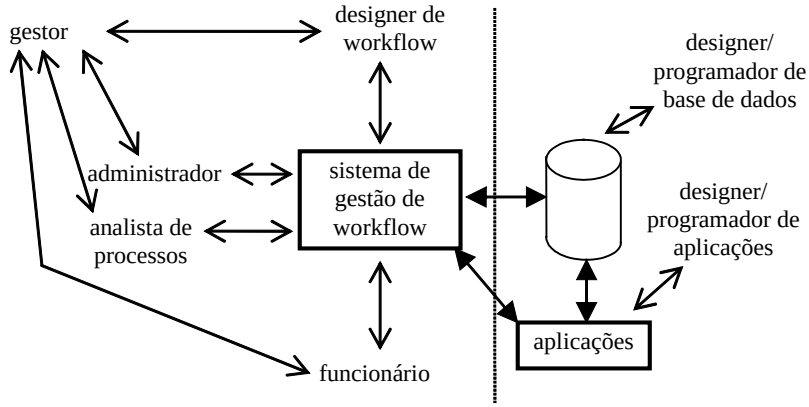


Figura 5.5. Os utilizadores de um sistema de gestão de *workflow*

5.3 Troca e Armazenamento de Dados

Um sistema de *workflow* integra um grande número de componentes. Para que todo o sistema opere correctamente, estes componentes necessitam de trocar informação entre si. Adicionalmente, é importante que diferentes tipos de dados sejam guardados. Com base na figura 5.4 iremos mostrar quais os dados que são administrados num sistema de *workflow*. Iremos depois examinar as ligações entre os vários componentes.

5.3.1 Dados num sistema de workflow

A figura 5.4 ilustra que os dados são de importância para o sistema de *workflow*. Na maior parte dos casos, o sistema de gestão de *workflow* e as aplicações fazem uso do mesmo sistema de base de dados. O sistema de *workflow*, então, "contrata" a administração dos dados a um sistema de gestão de base de dados. Os seguintes conjuntos de dados estão envolvidos:

1. *Definições de processos*. A definição de processos e tarefas. O nome, descrição, encaminhamento, tarefas e condições de cada processo são registados. Para cada tarefa, o seu nome, descrição, regras de decisão, conteúdo e regras de associação são registadas.

2. *Classificação de recursos*. A estruturação dos vários tipos de recursos. Para além de uma lista de classes de recursos (papéis ou unidades organizacionais), as relações entre elas são também registadas.

3. *Dados de análise*. Os resultados de todas as análises feitas. No caso das simulações, por exemplo, resultados de sub-simulações. (Por vezes uma simulação também utiliza dados históricos.)

4. *Dados de gestão operacional*. Os dados que são importantes para o administrador de um sistema de *workflow*. Por exemplo, a informação acerca da configuração técnica do sistema (parâmetros de sistema), informação sobre os funcionários e dados relacionados com os casos.

5. *Dados históricos*. Os dados que são guardados de forma a ser possível verificar o percurso de um caso específico, verificar a causa de um problema ou verificar o desempenho de um processo de negócio.

6. *Dados da aplicação*. Os dados que podem ser acedidos por uma aplicação mas não pelo sistema de gestão de *workflow*. Existem dois tipos de dados de aplicação: dados dos casos e dados de gestão. Os dados dos casos estão directamente relacionados com casos específicos. Os dados de gestão incluem informação geral sobre clientes e fornecedores.

7. *Dados internos*. Todos os dados que são mantidos pelo sistema de gestão de *workflow* mas que não se relacionam directamente com o *workflow*. Por exemplo, informação sobre listas de trabalho que estão activas, o estado de cada motor e endereços de rede. Ao contrário dos dados de gestão operacional, os dados internos são de natureza técnica e por conseguinte apenas acessíveis pelo serviço de execução.

8. *Dados de gestão logística*. O estado de cada *workflow* está embebido nos dados de gestão logística que consistem de informação sobre o estado dos casos (incluindo os atributos dos casos), o estado de cada recurso, e os disparos disponíveis. É preferível que estes estejam acessíveis apenas ao motor de *workflow*. Contudo, por razões técnicas, por vezes, é inevitável que estes sejam consultados e alterados por aplicações externas.

5.3.2 Problemas das interfaces

Um sistema de *workflow* consiste num grande número de componentes. Alguns deles são ferramentas de gestão do sistema de *workflow*, enquanto outros são as aplicações que suportam a execução das tarefas. Para essas componentes funcionarem em harmonia devem trocar informação. Portanto,

existem convenções que foram criadas pelo WFMC sobre a normalização das interfaces entre os componentes. Como ilustra a figura 5.6, o WFMC reconhece cinco interfaces.

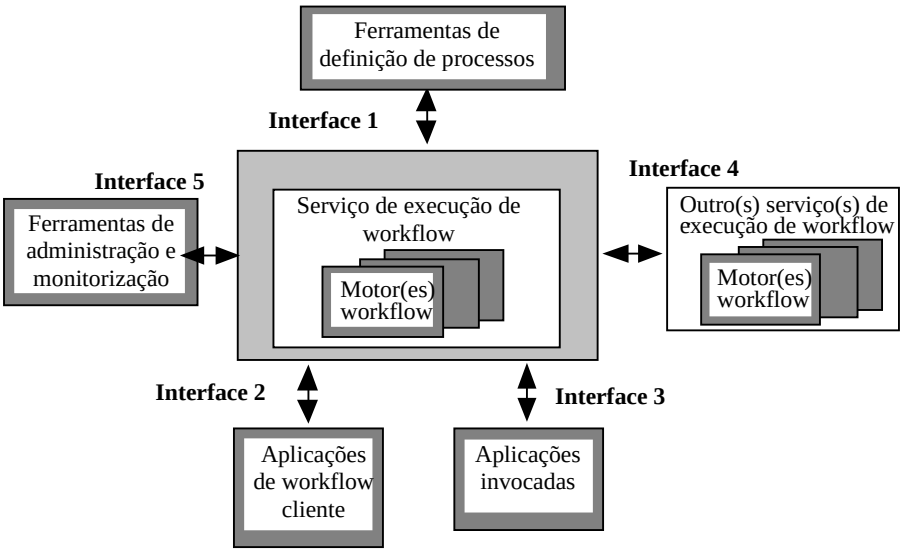


Figura 5.6. As interfaces entre vários elementos (©WFMC)

O objectivo da normalização das interfaces visa atingir três objectivos. Primeiro, os standards geralmente melhoram a troca de dados entre (partes dos) sistemas de gestão de *workflow*. Segundo, tornam possível a criação de ligações entre os serviços de execução de *workflow* desenvolvidos por diferentes fabricantes de forma simples. Finalmente, os standards permitem o desenvolvimento de aplicações que são inteiramente independentes do sistema de gestão de *workflow* escolhido.

Existe um número de interfaces que actualmente utilizam ficheiros e bases de dados. Por exemplo, na figura 5.4, assumimos que a Interface 1 e a Interface 5 eram implementadas usando uma base de dados. O WFMC, contudo, assume que todas as interfaces são implementadas utilizando uma API (*Application Programming Interface*). No contexto da gestão do *workflow*, o termo WAPI (*Workflow Application Programming Interface*) é também usado. Uma API é um grupo de *serviços* que são oferecidos a um *cliente* por um *servidor*. Estes serviços podem ser comparados com a chamada de procedimentos numa linguagem de programação convencional. A palavra cliente pode referir-se a uma aplicação. Um sistema operativo como o UNIX é um exemplo de um servidor. Podemos considerar a cópia de ficheiros como um serviço oferecido pelo sistema UNIX usando uma API (i.e., cp). No caso específico dos *workflows* (WAPI), o serviço de execução actua como um servidor e as aplicações e ferramentas actuam como clientes. Para fornecer um sumário das WAPIs reconhecidas pelo WFMC, descrevemos sucintamente o conteúdo de cada interface:

1. *Interface 1 (ferramentas de definição de processos)*. A Interface 1 determina a ligação entre as ferramentas para a criação e modificação da definição dos *workflows* e o serviço de execução dos processos de *workflow*. Esta WAPI contém funções para o início e o fecho de uma ligação (*connect/disconnect*), obtenção de resumos das definições do *workflow* (definição dos processos e classificações de recursos), e abertura, criação e registo da definição dos processos.

2. *Interface 2 (aplicações de workflow cliente)*. A segunda interface é dedicada à comunicação entre o serviço de execução de processos e o suporte às listas de tarefas. A WAPI que disponibiliza estas características suporta, entre outras, as seguintes funções: abertura e fecho de uma ligação, produção de sumários do estado dos casos e dos itens de trabalho, geração de novos casos e o início, interrupção e conclusão de actividades.

3. *Interface 3 (aplicações invocadas)*. As aplicações são invocadas pelo sistema de gestão de *workflow* através da Interface 3. A figura 5.6 sugere que todas as aplicações são iniciadas directamente pelo serviço de execução de processos de *workflow*, mas nem sempre isso acontece. Uma aplicação interactiva, como um processador de texto, é geralmente iniciada a partir de uma lista de tarefas.

4. *Interface 4 (outros serviços de execução de workflows)*. A Interface 4 disponibiliza a troca de trabalho entre vários sistemas de *workflow* autónomos (por exemplo, a transferência de argumentos e o *outsourcing* de itens de trabalho). Portanto, esta WAPI facilita a interoperabilidade de *workflows*.

5. *Interface 5 (ferramentas de administração e monitorização)*. A Interface 5 preocupa-se com a ligação entre ferramentas de administração e monitorização com o serviço de execução de *workflows*. A interface está subdividida em 2 partes: funções do sistema de gestão de *workflow* e funções de controlo no *workflow*. A primeira pode incluir a adição de um funcionário, a permissão de uma autorização e a execução da definição de um processo. Para controlar um *workflow*, o serviço de execução de processos guarda uma variedade de eventos num ficheiro *log*. Perguntas específicas sobre o histórico podem ser realizadas usando a Interface 5. Estas perguntas podem incluir tempos de espera, tempos de conclusão, tempos de processamento, encaminhamento e utilização dos recursos humanos.

O WFMC continua a trabalhar na normalização das WAPIs. Por exemplo, poucos progressos têm sido realizados sobre a normalização das Interfaces 3 e 5. Não obstante, a discussão das 5 interfaces fornece uma boa informação sobre as funcionalidades desejadas para um sistema de gestão de *workflow*.

Para aqueles envolvidos na introdução de um sistema de gestão de *workflow*, a Interface 3 é de particular importância. A Interface 4 torna-se somente significante quando alguém deseja interligar mais do que um sistema de *workflow*. A Interface 2 é útil quando o gestor de listas de tarefas *standard* já não é adequado e uma aplicação integrada necessita de ser desenvolvida. A interface 5 torna-se significante quando é necessário compilar informação de gestão sobre os eventos guardados pelo serviço de execução. Na prática, as

Interfaces 3 e 4 parecem causar mais problemas. Consequentemente, iremos considerar os seus potenciais problemas em mais detalhe.

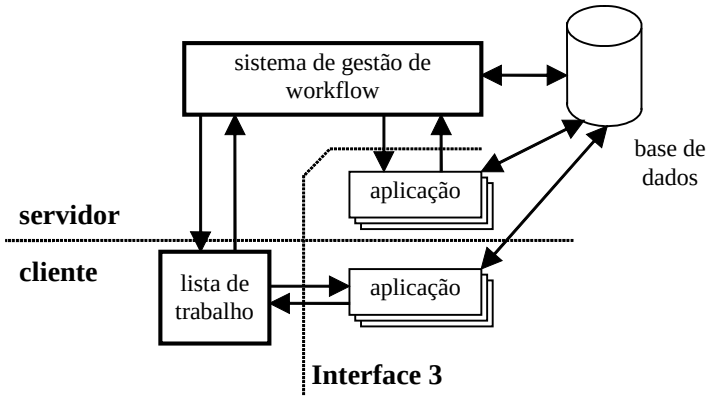


Figura 5.7. Potenciais problemas em torno da Interface 3

A figura 5.7 mostra através de um diagrama como pode ser iniciada uma aplicação (Interface 3). Isto pode ser feito pelo motor e/ou pelo gestor de listas de tarefas. Uma aplicação é chamada para executar uma tarefa específica. O motor inicia a execução da tarefa iniciando uma aplicação. Esta aplicação provavelmente irá modificar os dados da aplicação na base de dados. Se o motor de *workflow* não se encontrar disponível durante a execução da aplicação, devido a um erro de sistema, então o motor e a aplicação não estarão sincronizados. Uma vez que o sistema tenha sido corrigido, o motor não tem escolha senão "rebobinar" (repetir) a tarefa. Não há forma de saber se a aplicação concluiu a tarefa com sucesso e se as alterações aos atributos do caso foram transferidas para o sistema de *workflow*. Isto resulta numa incompatibilidade entre os dados logísticos (estado do caso) e os dados da aplicação. Assim, podem surgir consequências desastrosas. Consideremos, por exemplo, o pagamento realizado por um banco: se a aplicação realizou um pagamento, mas o sistema de gestão do *workflow* não está ciente disso, talvez devido a uma falha, então o pagamento poderá ser feito novamente.

Problemas semelhantes podem ocorrer quando a aplicação é iniciada através do gestor das listas de tarefas. Suponha-se que ocorreu um erro no gestor de tarefas enquanto a aplicação estava a ser executada. Novamente, o sistema de *workflow* e a aplicação voltam a estar dessincronizados. Pelo facto do motor de *workflow*, da base de dados, do gestor de tarefas e das aplicações poderem funcionar em sistemas diferentes torna estes problemas ainda mais significativos. Num ambiente cliente/servidor, por exemplo, o gestor de tarefas e parte da aplicação são executados localmente (no cliente), mas o restante funciona centralmente (servidor). Para resolver tais problemas efectivamente é vital que o motor de *workflow*, a base de dados, o gestor de tarefas e as aplicações considerem uma tarefa (ou parte da tarefa) como a *unidade lógica de trabalho comum*. Isto significa que se aplicam as propriedades *ACID* (atomicidade, consistência, isolamento e durabilidade) a este tipo de tarefas:

- *Atomicidade*. Qualquer tarefa é concluída com sucesso por inteiro (*commit*) ou recomeça a partir do início (*rollback*).
- *Consistência*. O resultado de uma actividade (por outras palavras, a execução de uma tarefa) conduz a um estado consistente.
- *Isolamento*. Quando diversas tarefas são realizadas em simultâneo o resultado é o mesmo do que se tivessem sido realizadas separadamente. Por outras palavras, as tarefas executadas ao mesmo tempo não se devem influenciar mutuamente. Esta propriedade também é referida por "serializabilidade".
- *Durabilidade*. A partir do momento em que uma tarefa tenha sido concluída com sucesso, o resultado deve ser guardado. Consequentemente, quando uma tarefa está completa deve garantir que o resultado não será perdido.

No *processamento transaccional*, como acontece no mundo financeiro, frequentemente temos de "passar o teste das propriedades ACID". Na prática, com a geração actual de sistemas de gestão de *workflow*, não é fácil garantir essas propriedades. Consequentemente, este aspecto deve ser considerado numa fase inicial.

Encontramos problemas similares quando interligamos dois ou mais sistemas de *workflow* (Interface 4). Além disso, na maioria dos sistemas de gestão de *workflow* nem sempre é inteiramente claro qual o estado dos casos. Em termos de redes de Petri, o estado de um caso corresponde à distribuição de testemunhos pelos lugares (condições) e pelos valores dos atributos do caso. A transferência de um caso entre dois sistemas de *workflow* baseados em redes de Petri é equivalente a transferir os testemunhos e atributos do caso. Em muitos outros sistemas de *workflow* a situação não é tão simples porque oferecem estados abstractos num nível conceptual para os casos. (Os lugares são omitidos da definição dos processos). Em tais casos, um trabalho complexo de "tradução" tem de ser realizado para transferir casos de um sistema para outro. Note-se que, adicionalmente à transferência de casos, o *outsourcing* dos itens de trabalho e a geração de novos casos num sistema diferente também pertencem ao domínio da interoperabilidade dos *workflows*.

5.3.3 Standards de interoperabilidade

Este capítulo baseia-se no modelo de referência do WFMC. Este modelo foi escolhido como ponto de partida, pois fornece uma introdução agradável à tecnologia de *workflow*. Muitos autores criticaram o modelo de referência por ser muito ingénuo ou por realçar assuntos pouco relevantes. Neste capítulo não iremos comparar o modelo de referência com arquitecturas alternativas, essas discussões técnicas estão fora do contexto deste livro. Contudo, iremos mostrar, neste capítulo, os recentes esforços realizados para resolver os problemas de interoperabilidade.

Nos últimos dois anos vários *standards* de interoperabilidade, isto é, especificações para a troca de informação entre produtos de *workflow*, foram propostos. Podemos classificar essas especificações de interoperabilidade em duas categorias: especificações para a modelação e descrição de *workflows* (tempo de desenho) e especificações para a interoperabilidade em tempo de execução (*runtime*).

A primeira categoria corresponde à Interface 1 do modelo de referência do WFMC. O WPDL (*WFMC Process Definition Language*) entra nesta categoria. Outro exemplo é o PIF (*Process Interchange Format*), um formato de intercâmbio desenhado para ajudar a troca automática de descrições de processos entre uma larga variedade de ferramentas de processos, como ferramentas de modelação de processos, sistemas de *workflow*, repositórios de processos, etc. Essas ferramentas podem inter-operar traduzindo o formato da descrição dos processos nativos para o formato PIF, e vice-versa. Desta forma, as descrições de processos podem ser trocadas automaticamente sem ser necessário utilizar diferentes tradutores entre sistemas. Se uma tradução do, ou para, PIF não pode ser realizada automaticamente, será necessário uma intervenção humana. O formato PIF ainda não ganhou suficiente ímpeto para se tornar um *standard* na indústria. Contudo, muitas ideias têm sido adoptadas por uma nova iniciativa: a linguagem de especificação de processos (*Process Specification Language – PSL*). A PSL é promovida pelo NIST (*U.S. National Institute of Standards and Technology*) e tem um âmbito maior que o WPDL do WFMC. Existem outros *standards* gerais que realçam diferentes aspectos, por exemplo, os esforços de normalização no contexto do UML (diagramas de estado, diagramas de sequencias, diagramas de colaboração, diagramas de actividade), o standard ISO para redes de Petri de altos nível (ISO/IEC JTC1/SC7/WG11), e também o bem conhecido standard IDEF0 (também suportado pelo NIST). Estes esforços de normalização são relevantes mas claramente não fornecem soluções para os problemas de interoperabilidade em tempo de desenho. Este é o resultado da ausência de um modelo conceptual comum ou núcleo formal, como foi mencionado anteriormente

A segunda categoria está centrada na interoperabilidade em tempo de execução. Esta categoria corresponde à Interface 2, Interface 3 e Interface 4, com foco na Interface 4. O ponto de ênfase está no suporte da troca de informação relativa à execução dos processos. Claramente, a Interface 4 tem um significado extremo quando é necessário trocar informação de execução entre sistemas desenvolvidos por diferentes vendedores. A mais notável iniciativa com respeito a interoperabilidade em tempo de execução foi a especificação de interoperabilidade da WFMC, o SWAP, o WF-XML, e o jointFlow da OMG. Já em 1996, o WFMC lançou a *Especificação de Interoperabilidade Abstracta* (WFMC-TC-1012). Isto foi seguido pelo *Interoperability Internet e-mail MIMI Binding* (WFMC-TC-1018). Recentemente (Maio de 2000), o WFMC lançou o *Interoperability Wf-XML Binding* (WFMC-TC-1023). Este último descreve uma realização de *Especificação de Interoperabilidade Abstracta* usando o XML e sendo esta baseada no SWAP. O SWAP (*Simple Workflow Access Protocol*) é um *standard* baseado na Internet e suporta diferentes vendedores de *workflows*. O SWAP utiliza o protocolo HTTP e pode ser usado para controlar e monitorizar processos de *workflow*. O jointFlow da OMG é uma iniciativa baseada numa arquitectura CORBA e também utilizada como ponto de partida a *Especificação de Interoperabilidade Abstracta* do WFMC. O jointFlow é formado por um conjunto de especificações IDL. O standard preocupa-se com a importância da interoperabilidade em tempo real para a realização de um sistema de *workflow*. No contexto do comércio electrónico, estes *standards* irão tornar-se ainda mais importantes. Infelizmente, os *standards* têm um cariz técnico e não lidam realmente com questões ao nível do negócio. É possível ligar sistemas de

diferentes vendedores usando, por exemplo, o Wf-XML. Todavia, isto não implica que o processo seja executado como era desejado.

5.4 Infra-estrutura Técnica Necessária

Para conseguir desenvolver um sistema de *workflow* funcional, não é suficiente adquirir um sistema de gestão do *workflow*. Como mostra a figura 5.8, esse é somente um dos componentes requeridos.

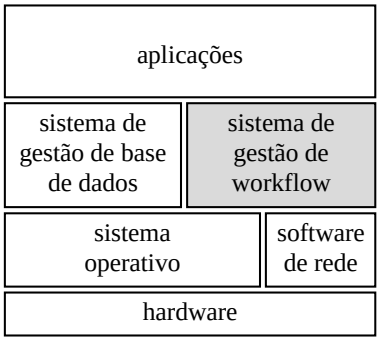


Figura 5.8. Um sumário dos componentes técnicos

O sucesso da instalação de um sistema de *workflow* depende da existência de uma infra-estrutura técnica adequada. A maioria opera num ambiente cliente/servidor. Tal ambiente consiste tipicamente num servidor central que funciona no Windows NT/2000 ou no UNIX e um número de clientes que usam o MS-DOS/Windows 3.1, o OS/2 ou o Windows 95/98/2000. Como já tínhamos visto na figura 5.7, o motor de *workflow* funciona do lado do servidor. O gestor das listas de tarefas, e por conseguinte, a interface com o utilizador, opera do lado do cliente. As aplicações podem operar em qualquer um dos lados. A base de dados da gestão e os dados da aplicação são administrados pelo servidor. Sem querer dar uma explicação muito técnica, iremos brevemente descrever os principais componentes:

1. *Hardware*. O servidor é usualmente um microcomputador poderoso, um minicomputador ou um mainframe. Os computadores com um conjunto reduzido de instruções (RISC) são muitas vezes usados. Os clientes geralmente escolhem computadores com um conjunto de instruções complexo (CISC): por exemplo, computadores pessoais (PC) baseados em processadores Intel 80x86. O servidor é ligado ao cliente usando cabos coaxiais, cabos de fibra óptica, ou pares entrelaçados com ou sem protecção. *Bridges*, *routers*, *hubs* e/ou *gateways* também podem ser necessários quando construímos grandes redes.

2. *Sistemas operativos*. O sistema operativo do servidor deve ser multi-utilizador e multi-tarefa. Uma escolha óbvia é o UNIX; outras possibilidades são o OS/2, o Windows NT/2000, ou o Linux. Os *mainframes* são raramente usados para a gestão de *workflows*. Sistemas operativos como o VMS, o MVS e o AS400 são

também raramente suportados pela geração actual de sistemas de gestão de *workflows*. O sistema operativo do cliente é geralmente o Windows 95/98/2000. Contudo, também é utilizado o sistema UNIX, o OS/2, ou o Linux. Uma característica dos sistemas operativos modernos é que todos eles suportam a interface com o utilizador.

3. *Software de rede*. A rede desempenha um papel crucial na operação de um sistema de *workflow*. Ela liga os clientes ao servidor. As escolhas mais comuns da tecnologia de rede são a Ethernet e o protocolo Token Ring. O *software* de comunicação utiliza um protocolo semelhante para a troca de mensagens. O TCP/IP (Transmission Control Protocol/Internet Protocol) é actualmente o *standard* mais usado num ambiente cliente/servidor. Outras possibilidades incluem o NetWare, o SNA e o AppleTalk.

4. *Sistemas de gestão da base de dados*. Muitos sistemas de informação são construídos à volta de um sistema de base de dados. Num sistema de *workflow* a base de dados também desempenha um papel importante. Usualmente, as aplicações e o sistema de gestão de *workflow* usam o mesmo sistema de base de dados. Isto significa que o sistema de gestão de *workflow* deve poder utilizar o sistema de gestão da base de dados que já tinha sido escolhido. Portanto, a maioria dos sistemas de gestão de *workflow* suporta um vasto conjunto de base de dados relacionais tais como o Oracle, o Sybase e o SQLserver. Utilizando o ODBC (*Open DataBase Connectivity*), é, em teoria, possível tornar o sistema de gestão de *workflow* independente do sistema de gestão subjacente à base de dados. Contudo, a selecção de uma combinação incompatível pode resultar num pobre desempenho de todo o sistema de *workflow*.

5. *Aplicações*. As aplicações suportam a execução das tarefas. Estas podem consistir de aplicações de *software standard*, como um processador de texto, uma folha de cálculo, ou um *software* geralmente escrito numa linguagem de programação, tal como *script*, uma linguagem de terceira geração (como o C++ ou o Java), ou uma linguagem de quarta geração (como o Powerbuilder ou o Oracle Designer/2000). Vários mecanismos são concebíveis para dar início a uma aplicação. A linha de comandos pode ser usada (por outras palavras, a aplicação é iniciada directamente a partir do sistema operativo). Os atributos de um caso podem ser trocados através da WAPI ou pela base de dados. O inconveniente é que um novo programa deve ser iniciado para cada actividade. Consequentemente, por vezes, é melhor iniciar a aplicação somente uma vez. Em tais casos, a aplicação não é encerrada quando a actividade é concluída. Então, iniciar a aplicação uma segunda, terceira, ou quarta vez já não é necessário. No Windows, por exemplo, o DDE (*Dynamic Data Exchange*) é usado para conseguir isso.

6. *Sistemas de gestão de workflow*. O sistema de gestão de *workflow* tem de lidar com cada um dos componentes referidos anteriormente. Deve poder trocar informação com as aplicações e com o sistema de base de dados. Além disso, deve poder lidar eficientemente com a capacidade disponível de processamento e da rede.

Os pontos anteriormente descritos mostram que os aspectos técnicos e os aspectos funcionais necessitam de ser tomados em conta durante a aquisição de um sistema de gestão de *workflow*. Tal sistema utiliza *hardware*, sistemas operativos, software de rede, sistemas de base de dados e aplicações já existentes. Consequentemente, é vital que o sistema de gestão de *workflow* escolhido seja compatível com aqueles componentes. Uma pobre combinação pode resultar num sistema pouco fiável com um grande tempo de resposta e uma baixa velocidade de processamento.

5.5 Geração Actual dos Produtos de Workflow

Actualmente, estão disponíveis vários sistemas de *workflow* no mercado. A figura 5.9 lista alguns deles e está muito longe de estar completa. Na realidade, o suporte de alguns dos produtos listados já não existe. O número de fornecedores que oferece *software* de gestão de *workflow* está estimado em duas centenas – o que indica que se espera que tais sistemas venham a ter um papel mais importante num futuro próximo. Além dos sistemas de gestão de *workflow* especializados, os sistemas ERP, tal como o SAP, o Baan e o JD Edwards têm um motor de *workflow* incorporado. Na maioria dos casos esses motores de *workflow* não podem ser utilizados como sistemas de gestão de *workflow* individualmente e autonomamente.

A informação deste capítulo tem por base a situação verificada no início do ano 2000. Devido ao elevado ritmo de desenvolvimento no mercado do *workflow*, este levantamento estará provavelmente desactualizado dentro de alguns anos. O resto deste livro é, no entanto, menos dependente do tempo, e continuará actual durante muitos anos.

Apesar do grande número de fornecedores, alguns dos quais listados na figura 5.9, o número de sistemas de *workflow* actualmente em produção é relativamente reduzido. Há diversas razões para isto. Primeiro, a tecnologia é completamente nova, também os construtores destes sistemas têm um conhecimento insuficiente das possibilidades oferecidas pelos sistemas de gestão de *workflow*. Também, muitos sistemas de gestão de *workflow* ainda não estão completamente desenvolvidos, resultando em funcionalidades limitadas e oferecendo uma fiabilidade insatisfatória. Presentemente, não é fácil optar por um sistema de gestão de *workflow* específico. O grande número de sistemas disponíveis e o alto grau de incerteza sobre o futuro torna a escolha ainda mais difícil. Finalmente, apesar dos esforços do WFMC, há uma carência de *standards* que descrevam as funcionalidades e as interligações entre sistemas. Por exemplo, muitos sistemas de gestão de *workflow* usam uma técnica de desenho *ad hoc* para processos específicos. Um dos inconvenientes desta aproximação é a dificuldade da troca de descrições de processos entre diferentes fornecedores de sistemas (um *standard* conceptual baseado em redes de Petri iria trazer um contributo significante nesta área). Apesar destes obstáculos, a importância da gestão de *workflow* irá aumentar no futuro.

Para ganhar uma impressão da geração actual dos sistemas de gestão de *workflow*, iremos examinar três produtos: Staffware® (Staffware Plc), COSA® (Ley GmbH), e ActionWorkflow® (Action Technologies Inc.). O Staffware é um dos produtos líderes com uma quota de mercado estimada em 25%, portanto é um bom exemplo das capacidades dos sistemas de gestão de *workflow* actuais.

Os últimos dois produtos foram escolhidos porque representam extremos no espectro dos sistemas de gestão de *workflow*. O COSA é um produto robusto com extensas possibilidades para gerir processos de negócio complexos. Também segue minuciosamente a técnica de modelação de processos usada neste livro. O ActionWorkflow representa uma abordagem completamente diferente em que é dado um ênfase maior na coordenação das partes envolvidas em detrimento da gestão de processos. O Staffware será discutido com maior detalhe. Os outros dois sistemas serão apenas brevemente discutidos. Iremos mostrar algumas ferramentas de análise de *workflow* e de BPR e mencionar alguns critérios que podem ser usados para a selecção de um sistema de gestão de *workflow*.

ActionWorkflow	Action Technologies Inc.
Computron Workflow	Computron
COSA	Ley GmbH
CSE/WorkFlow	CSE
Documetrix Workflow	Universal Systems Inc.
FloWare	BancTec-Plexus
FLOWBuilder	PowerCerv
FlowMark/MQ Series Workflow	IBM
FormFlow	Delrina
HICOS	Empirica
InConcert	TIBCO/InConcert
Income	Promatis
JetForm Server	JetForm Corporation
KI Shell	UES Inc.
NAVIGATOR 2000/Workflow	I. Levy & Associates
Open Workflow	Wang
OPEN IMAGE	SNS Systems
PowerFlow	Optika Imaging Systems Inc.
Process Weaver	Cap Gemini Innovation
SAP Business Workflow	SAP AG
Staffware	Staffware
TeamWARE	TeamWARE
Ultimus	Ultimus
Verve	Verve Inc.
ViewStar	ViewStar
Visual WorkFlo	FileNet Corp.
WebFlow	Cap Gemini Innovation
Workflow Factory	Delphi Consulting Group
WorkFLOW SQL	Optical Image Technology Inc.
WorkParty	Siemens Nixdorf IS-AG
WorkVision	IA Corporation

Figura 5.9. Um conjunto de sistemas de *workflow* e seus fornecedores

5.5.1 Staffware

O Staffware® é um dos sistemas de gestão de *workflow* mais divulgados em todo o mundo. Em 1998 foi estimado pelo Gartner Group que o Staffware dominava mais de 25% do mercado global. A empresa, Staffware Plc, que desenvolve e distribui o Staffware está situada na cidade de Maidenhead, no Reino Unido. Nesta secção iremos descrever a versão actual do Staffware, nomeadamente, o Staffware 2000, o sucessor do Staffware 97, que foi lançado nos finais de 1999.

O Staffware é constituído pelos seguintes componentes:

1. *Graphical Workflow Definer (GWD)*. O GWD é a ferramenta de definição de processos do Staffware. No entanto, não suporta qualquer forma de análise.

2. *Graphical Form Designer (GFD)*. O GFD é usado para definir a interface que é apresentada ao utilizador final ou, no caso de tarefas automáticas, a interface que é apresentada às aplicações externas.

3. *Work Queue Manager (WQM)*. O WQM é a ferramenta cliente do Staffware que é usada para oferecer itens de trabalho aos utilizadores finais.

4. *Staffware Server (SS)*. O servidor do Staffware é responsável pela execução dos workflows.

5. *Staffware Administration Managers (SAM)*. O SAM consiste num conjunto de ferramentas que fornecem suporte às funções dos administradores dos workflows. Nele estão incluídas as seguintes ferramentas: *user manager, backup manager, table manager, case manager, list manager, network manager e sysinfo.*

6. *Audit Trail (AT)*. O AT é usado para monitorizar a execução de casos individuais.

Os componentes do Staffware podem ser relacionados muito facilmente com o modelo de referência do WFMC (Workflow Management Coalition): o GWD e o GFD correspondem às ferramentas de definição de processos (Interface 1), o WQM corresponde à aplicação cliente de *workflow* (Interface 2), o SAM e o AT correspondem às ferramentas de administração e monitorização (Interface 5) e o SS fornece o serviço de execução do Staffware.

A figura 5.10 mostra o GWD. A linguagem de modelação usada é específica do Staffware. As tarefas são chamadas de *etapas* (*steps*). Existem vários tipos de etapas: etapas automáticas (oferecidas a uma aplicação e não a um utilizador final), etapas normais (executadas por utilizadores), e etapas de acontecimento (activadas por acontecimentos externos). A semântica das etapas são *OR-join/AND-split*, o que significa que *uma* etapa pode ficar pronta a ser executada se uma das tarefas anteriores for concluída com sucesso. A conclusão dessa etapa irá despoletar um disparo que será enviado a *todas* as etapas seguintes. Porque a semântica do OR-join/AND-split é fixa, dois blocos construtores adicionais são necessários: *etapas de espera* (*wait step*) e a *condição*. As etapas de espera podem ser usadas para sincronizar fluxos e têm a semântica OR-

join/AND-split. Para modelar escolhas, isto é, OR-splits, o bloco construtor que inclui uma condição pode ser usado. O Staffware permite apenas dois resultados possíveis (por exemplo, SIM e NÃO). Os processos do Staffware iniciam-se sempre com a etapa de início que é representada por um semáforo. A conclusão dos processos em Staffware é implícita; é possível iniciar múltiplos *threads* paralelos que se completam concorrentemente. Deste modo, não é necessário existir um nó de sincronização para representar a conclusão de um caso. O fim de cada *thread* é denotado pelo símbolo *stop*. As condições são modeladas por losangos. As tarefas de espera são modeladas por símbolos em forma de ampulheta. As semânticas básicas de uma etapa, condição e espera são ilustradas na figura 5.11.

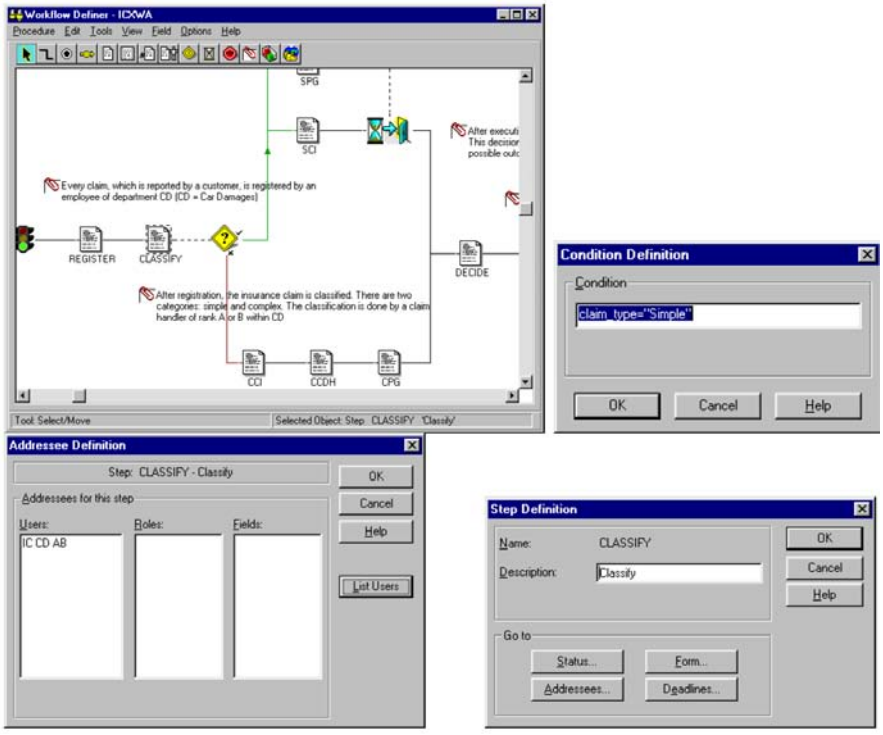


Figura 5.10. Graphical Workflow Definer (GWD): Ferramenta de desenho do Staffware

As associações ilustradas na figura 5.11 não consideram duas características adicionais disponíveis para as etapas. Em primeiro lugar, é possível remover etapas. Em segundo lugar, é possível modelar um *time-out*, ou seja, especificar uma etapa que irá desencadear outras etapas se não for executada dentro de um determinado período de tempo.

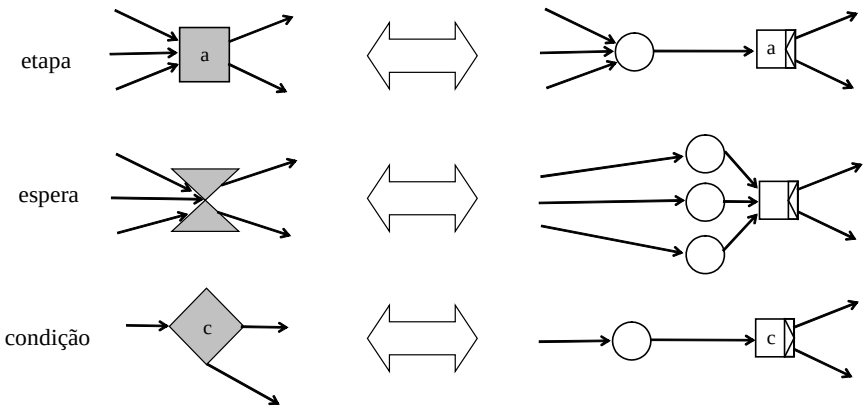


Figura 5.11. A semântica de algumas construções em Staffware (esquerda) expressas usando redes de Petri (direita)

A figura 5.12 mostra o processo de reivindicação de um seguro usado no capítulo 2 e modelado com o Staffware GWD. As figuras 5.11 e 5.12 mostram que a linguagem de modelação do Staffware é bastante semelhante às técnicas usadas ao longo deste livro: conceitos tais como AND/OR-split/join são regras importantes em ambos os modelos. Todavia, existem pequenas diferenças que são relevantes. Uma das diferenças é a noção de estado, ou seja, um conceito semelhante aos lugares das redes de Petri, não é suportado pelo Staffware. Como resultado, alguns modelos podem parecer mais simples em Staffware (por exemplo, um processo sequencial simples). Entretanto, outros modelos tornam-se maiores como consequência das escolhas binárias e da necessidade de introduzir etapas de espera com fins de sincronização. Na realidade, algumas construções que podem ser modeladas em redes de Petri não podem ser modeladas em Staffware, tais como, as escolhas implícitas, os *milestones* e outros elementos pertencentes à classe das redes de Petri de escolha não livre (*non-free-choice*). A única maneira de suportar estas construções é codificar as funcionalidades numa aplicação externa ou aceitar uma semântica diferente.

O Staffware não disponibiliza uma ferramenta para a modelação organizacional. Como alternativa, utiliza o conceito de *filas de trabalho* (*work queue*). Uma fila de trabalho pode ser comparada com uma classe de recursos. Toda a fila está associada a um *grupo de utilizadores*. Um utilizador pode ser membro de várias filas de trabalho e uma fila de trabalho pode estar associada a vários utilizadores. Cada utilizador vê as filas de trabalho às quais está associado. Um item de trabalho pode ser atribuído a uma ou mais filas de trabalho. Se um item de trabalho é atribuído a uma fila de trabalho, um elemento do grupo tem de executá-lo. Quando um utilizador quer executar um item de trabalho, selecciona-o da respectiva fila. Enquanto um utilizador executa um item de trabalho, o item fica bloqueado para os outros elementos do grupo. Depois do processamento, o utilizador pode libertá-lo (isto é, informa o sistema que o item já foi processado) ou pode voltar a colocá-lo na fila.

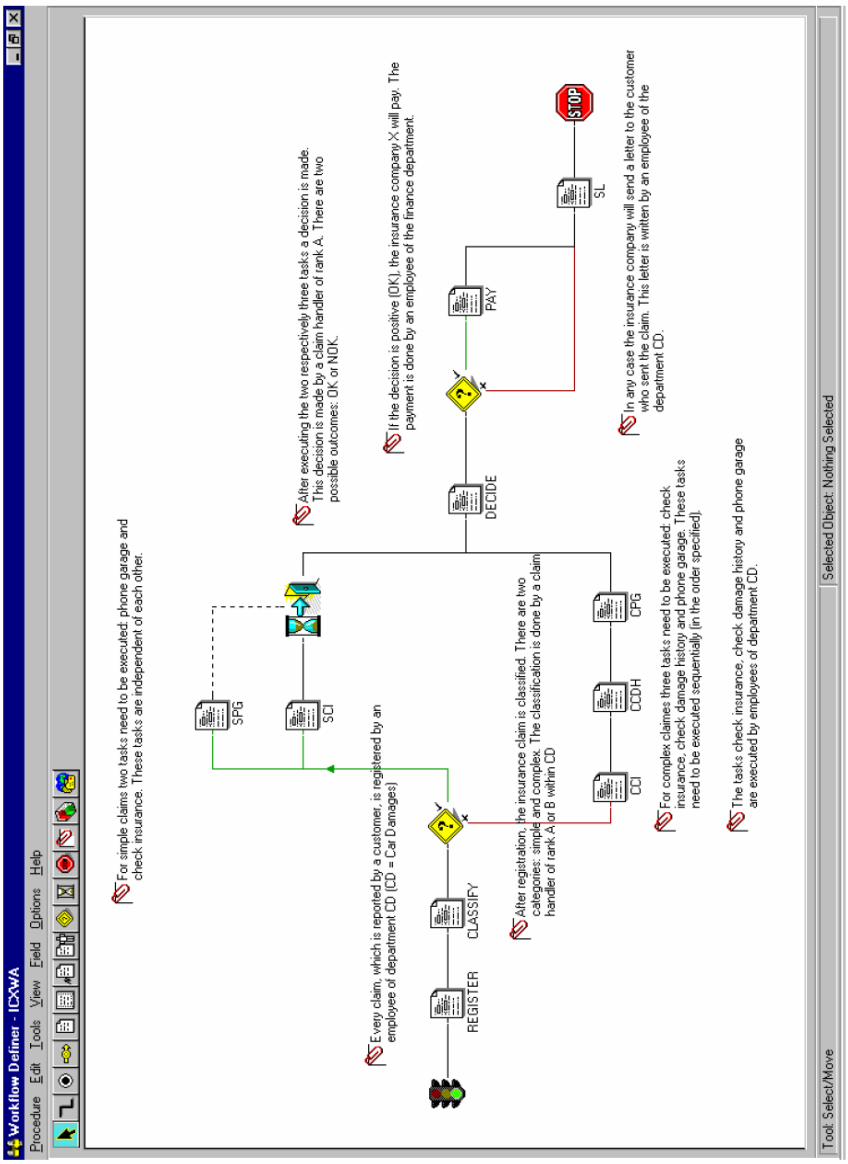


Figura 5.12. Um processo de reivindicação de um seguro modelado com o Staffware

A figura 5.13 mostra o WQM do Staffware. Esta ferramenta é usada para disponibilizar o trabalho ao utilizador final. Do lado esquerdo são ilustradas as filas de trabalho. Note-se que cada utilizador tem uma fila de trabalho pessoal e diversas filas de grupo. A figura 5.13 mostra quatro filas de grupo. Do lado direito, alguns dos itens de trabalho estão ilustrados. Seleccionando uma fila específica, o utilizador pode ver todos os itens associados a essa fila. Na figura 5.13 existem três itens de trabalho associados à fila de trabalho *IC CD Employee*.

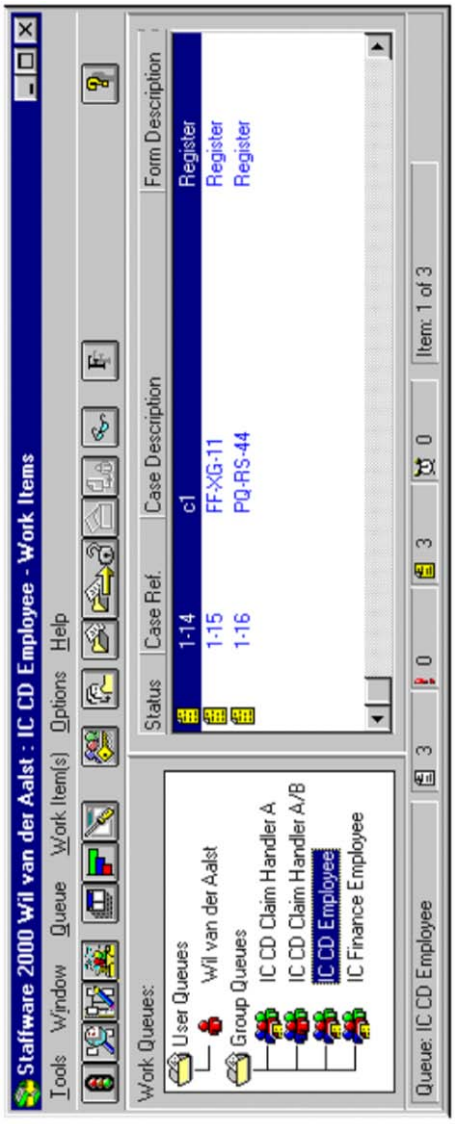


Figura 5.13. Work Queue Manager (WQM) do Staffware

A figura 5.14 mostra algumas ferramentas disponibilizadas pelo Staffware 2000. A ferramenta Audit Trail (topo do lado direito) mostra o conjunto de todas as ocorrências para um dado caso ou processo. O User Manager (em baixo) é usado para manter a lista dos utilizadores finais, os seus privilégios, as inscrições nas filas, etc. O User Manager é apenas uma das ferramentas do Staffware Administration Manager (SAM).

Concluímos, assim, a nossa introdução ao Staffware 2000. Foram ilustradas as principais características da actual geração de sistemas de gestão de *workflows*. A

descrição de outros dois sistemas de gestão de *workflows* (COSA e ActionWorkflow) será feita de maneira menos elaborada.

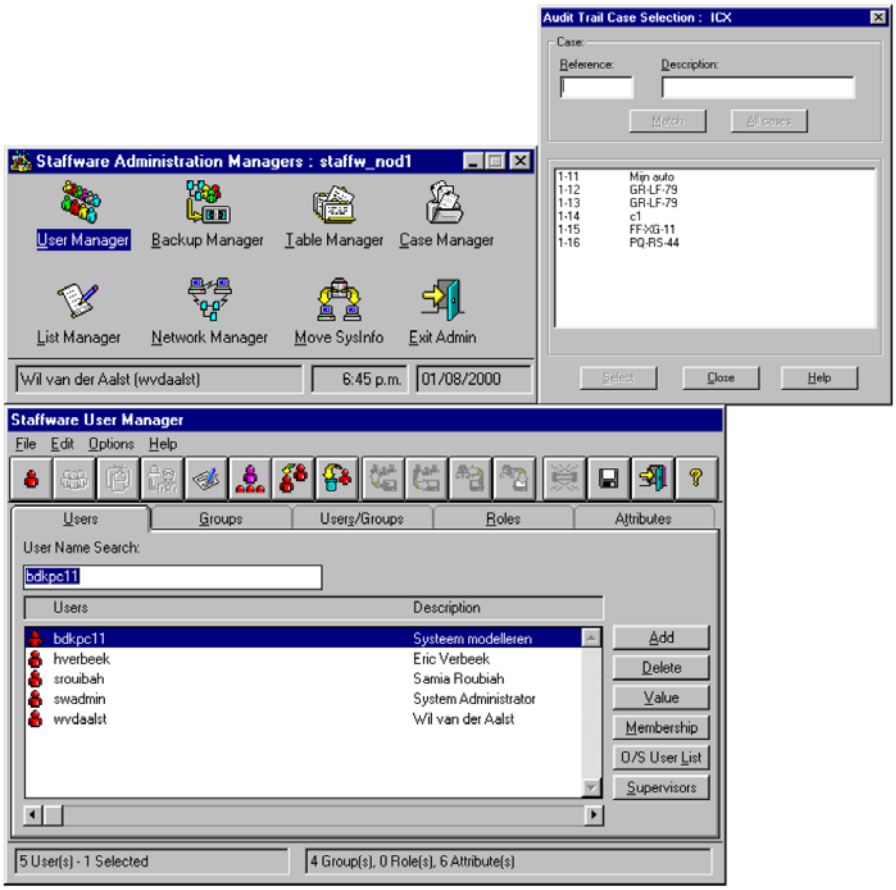


Figura 5.14. O Audit Trail e o User Manager (uma das ferramentas do Staffware para a gestão e administração)

5.5.2 COSA

O COSA® (COmputerunterstütze SAchbearbeitung) é produzido pelo Software-Ley GmbH. É um sistema de gestão de *workflow* baseado em redes de Petri. O COSA pode ser descrito como um sistema de gestão de *workflow* tradicional que segue o modelo de referência do WFMC. É também caracterizado pela vasta quantidade de funcionalidades e por possuir uma interface um tanto ou quanto desactualizada. As figuras exibidas nesta secção ilustram o COSA 1.4. A interface do utilizador do COSA 2.0 e a interface do COSA 3.0, que foi lançado recentemente, parecem completamente diferentes, mas na essência disponibilizam as mesmas funcionalidades.

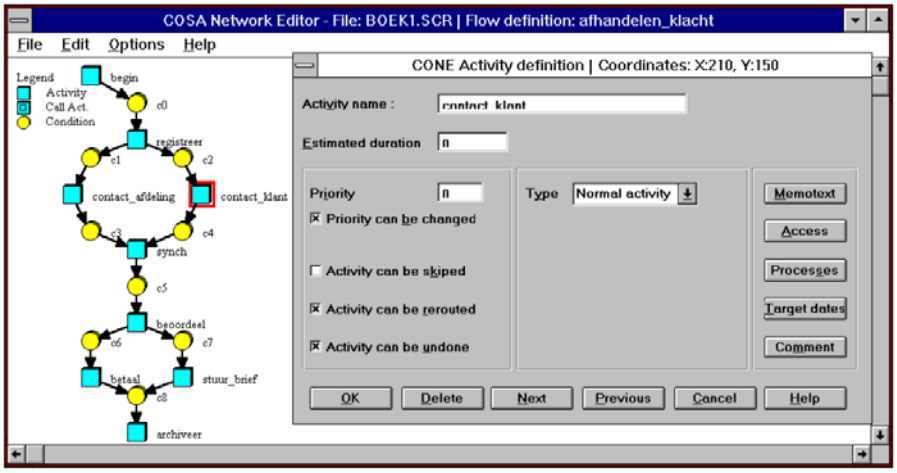


Figura 5.15. Definição de um processo utilizando o CONE

O COSA é constituído pelos seguintes componentes:

1. *COSA Network Editor (CONE)*. O CONE é uma ferramenta para definir e rever processos. Como ilustrado na figura 5.15, as redes de Petri são usadas para modelar os processos.

2. *COSA User Editor (COUE)*. O COUE é uma ferramenta de classificação de recursos para definir papéis e unidades organizacionais. A figura 5.16 mostra como as classes de recursos podem ser estruturadas hierarquicamente.

3. *COSA MemoBox (COMB)*. O COMB é uma ferramenta típica de gestão de listas de tarefas (worklist) para disponibilizar e iniciar os itens de trabalho (ver figura 5.3). É atribuída uma lista de trabalho pessoal a todos os funcionários.

4. *COSA Networkstate Displayer (COND)*. O COND é uma ferramenta gráfica para apresentar o estado de um caso. Porque um funcionário pode ver e analisar o estado de um caso, ele tem conhecimento do estado do processo de negócio.

5. *COSA Runtime Server (CORS)*. O CORS é o serviço de execução de workflow que consiste num ou mais motores.

6. *COSA Simulator (COSI)*. O COSA disponibiliza uma ferramenta primitiva para a simulação de processos de negócio. Existe também uma ligação disponível entre o COSA e a ferramenta de análise ExSpect.

7. *COSA Administrator (COAD)*. O COAD é usado para gerir os workflows. O COSA não disponibiliza ferramentas de registo nem ferramentas de relatório. Contudo, as ferramentas *standard* de geração de relatórios (tais como as ferramentas de Sistemas de Gestão de Informação, sistemas OLAP e sistemas de

extracção) podem ler e processar a informação necessária a partir da base de dados do COSA.

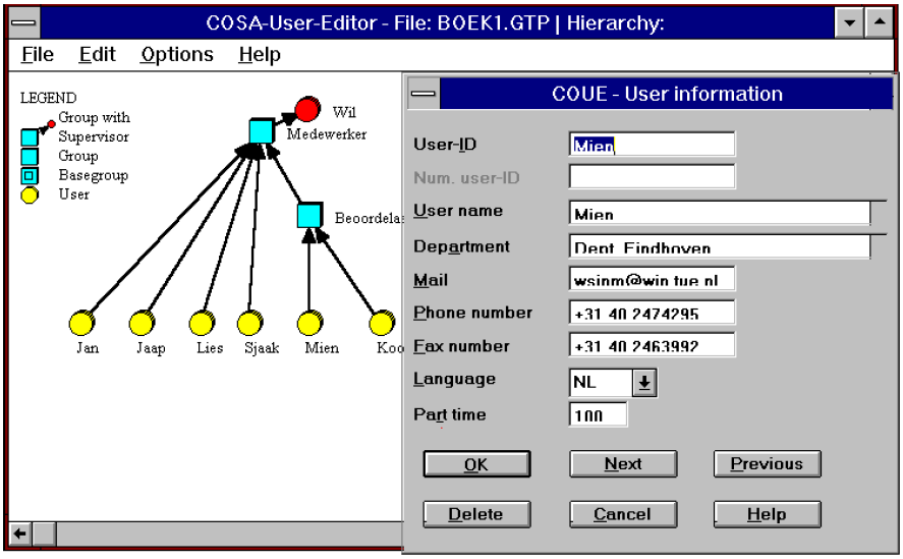


Figura 5.16. A subdivisão em papéis produzida pela COUE

A arquitectura do COSA pode ser facilmente relacionada com o modelo de referência do WFMC (ver figura 5.2). O CONE, o COUE e o COSI constituem as ferramentas de definição de processos (Interface 1). O COMB e o COAD correspondem, respectivamente, às aplicações clientes de *workflow* (Interface 2) e às ferramentas de administração e monitorização (Interface 5). O COND pode ser considerado como um complemento ao COMB.

O COSA suporta várias plataformas, UNIX, Windows NT/2000 e OS/2 no lado do servidor e OS/2, Windows NT/2000, Windows 3.1, Windows95/98/2000 e UNIX no lado do cliente. Os seguintes sistemas de gestão de base de dados são suportados: Oracle, Informix, Sybase, Ingres e DB2. É possível comunicar com *workflows* existentes na Internet usando o portal COSA, isto é, é possível aceder às funcionalidades da MemoBox através de um *Web browser*.

5.5.3 ActionWorkflow

O ActionWorkflow® é produzido pela Action Technologies Inc. e centra-se no suporte de processos no qual a comunicação entre pessoas e/ou equipas desempenha um papel fundamental. Neste sentido, o ActionWorkflow é bastante diferente dos sistemas de gestão de *workflow* tradicionais, como o COSA e o Staffware. Ao contrário do COSA e do Staffware, os quais se concentram em processos, o ActionWorkflow centra-se na coordenação. O ActionWorkflow usa os chamados *Business Process Maps* (BPM). Estes são construídos a partir de um número de *workflows* (ver figura 5.17). Cada *workflow* corresponde a uma

transacção que percorre os seguintes passos: (1) *preparação*, (2) *negociação*, (3) *desempenho* e (4) *conclusão*. As transições entre estados ocorrem usando os chamados *speech acts* (comunicação entre as pessoas/equipes envolvidas em transacções). Os *workflows* podem ser interligados entre si para ilustrar as ligações entre as transacções. Desta forma, diversos tipos de encaminhamento podem ser definidos. No BPM ilustrado na figura 5.17, os *workflows* D e E são executados em paralelo. O *workflow* C é executado após o *workflow* B.

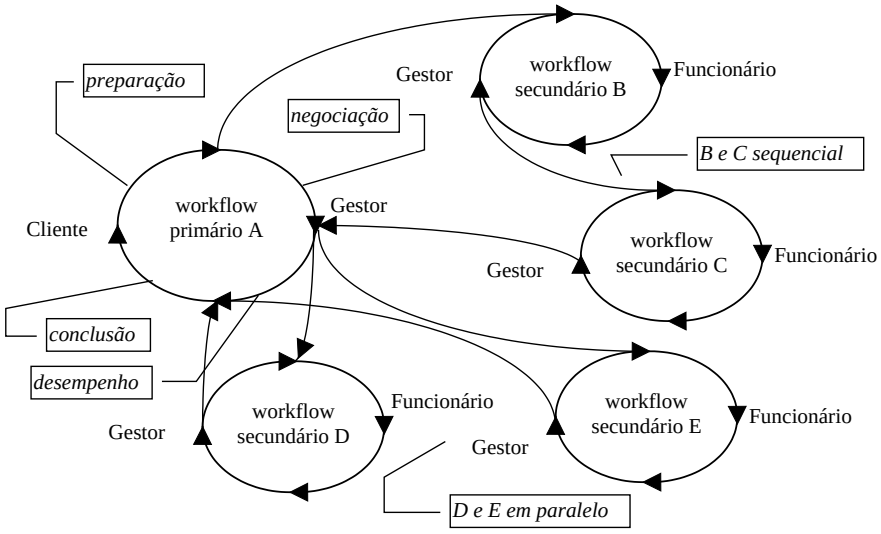


Figura 5.17. *Business Process Map* com um *workflow* primário e quarto *workflows* secundários

Nesta secção iremos discutir as funcionalidades do ActionWorkflow 3.0. Este não é o actual produto de *workflow* da empresa Action Technologies Inc. O foco da empresa Action Technologies Inc. mudou, da simples gestão de *workflow*, para soluções de negócio completas. Contudo, o seu produto mais recente chamado Action Works Metro (também conhecido por "e-process application platform") inclui as funcionalidades do ActionWorkflow 3.0.

O ActionWorkflow 3.0 também conhecido como ActionWorkflow Enterprise Series consiste nos seguintes componentes:

1. *Action Workflow Process Builder*. O Process Builder é usado para ilustrar *workflows* com a ajuda de *Business Process Maps*. Existem duas versões: a Analyst Edition, para o desenho de processo e a Developer Edition, para a actual realização.

2. *Action Workflow Process Manager*. O Process Manager é a essência do ActionWorkflow. É simultaneamente um motor de *workflow* e uma ferramenta para gerir *workflows*. Além disso, oferece possibilidades avançadas para analisar os *workflows* que estão em progresso.

3. *Action DocRoute*. O DocRoute é baseado no Process Manager e oferece a capacidade de integrar um gestor de documentos e aplicações de imagem.

4. *Action Metro*. O Action Metro proporciona a possibilidade de criar sistemas de *workflow* que serão usados na Internet. Web browsers tais como o Netscape Navigator e o Microsoft Internet Explorer podem ser usados como gestores de listas de trabalho.

Podemos ilustrar os componentes do ActionWorkflow usando o modelo de referência do WFMC. O ActionWorkflow Process Builder é a única ferramenta de definição de processos (Interface 1). O ActionWorkflow Process Manager corresponde ao serviço de execução de *workflows*, às ferramentas de administração e monitorização (Interface 5) e a uma parte das aplicações clientes do *workflow* (Interface 2). O Action DocRoute é difícil de situar no modelo de referência. O Action Metro pode ser considerado como alternativa à interface 2; um Web browser comporta-se como uma aplicação de *workflow* cliente.

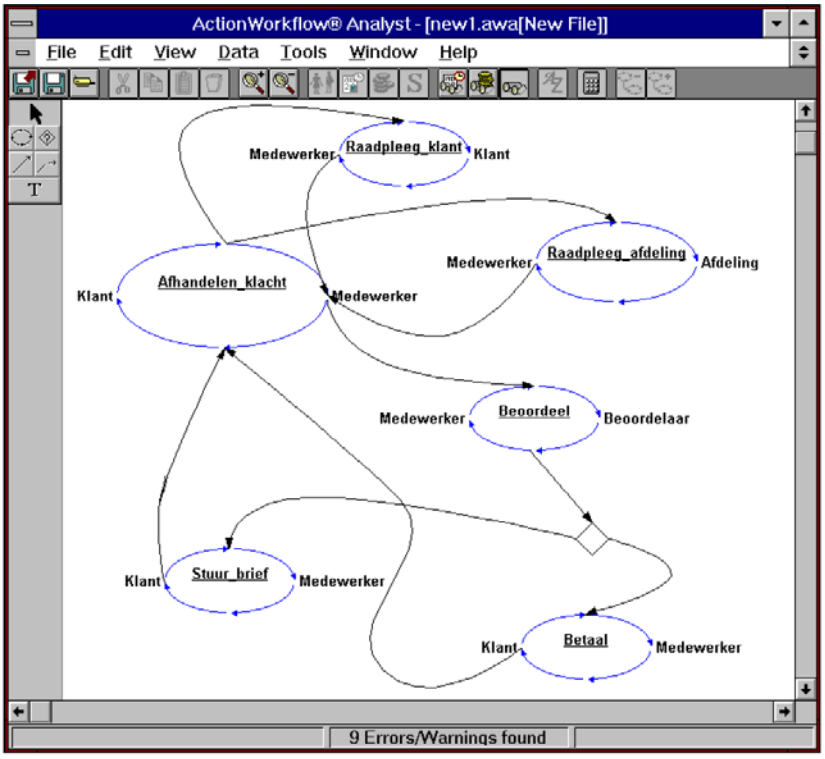


Figura 5.18. *Business Process Map* (BPM) construído usando o ActionWorkflow 2.0

O ActionWorkflow está apenas disponível para um número limitado de plataformas. O ActionWorkflow 3.0 está disponível apenas para o Windows NT/2000 no lado do servidor. O Process Builder opera também sob o Windows

95/98/2000. Através do uso da Internet, o *software* cliente é apropriado para quase todos os sistemas. A gestão de dados usa o Microsoft SQL Server.

A descrição anterior mostra que o COSA (ou o Staffware) e o ActionWorkflow são dois sistemas de gestão de *workflow* muito diferentes. O COSA é tradicional e completo, permitindo o suporte da maior parte dos processos de produção dentro de organizações administrativas. O ActionWorkflow difere em muitos aspectos dos sistemas de gestão de *workflow*, e parece ser mais adequado no suporte de processos no qual a coordenação é crucial.

5.5.4 Ferramentas de análise

Como foi apontado no capítulo anterior, existem diversas técnicas para a análise de sistemas de *workflow*. Infelizmente, os sistemas de gestão de *workflow* actuais dificilmente suportam qualquer tipo de análise. No capítulo 4 diferenciamos análises qualitativas (preocupadas em determinar se os processos estão correctos) de análises quantitativas (preocupadas com o desempenho e requisitos de capacidade). Apenas algumas ferramentas de *workflow* têm um foco na análise qualitativa. A maior parte dos sistemas de gestão de *workflow* têm unicamente verificações triviais de correcção, tal como: está o grafo de *workflow* totalmente ligado? Verificações mais avançadas, tais como a ausências de deadlocks, terminação garantida e terminação correcta, não são suportadas. Algumas ferramentas de pesquisa foram desenvolvidas para lidar com o problema da análise qualitativa. As mais notáveis incluem o *Woflan* (SMIS/I&T, Eindhoven University of Technology, The Netherlands) e *FlowMake* (DSTC Pty Ltd, The University of Queensland, Australia). Ambas as ferramentas são capazes de analisar propriedades semelhantes à propriedade de correcção definida no capítulo 4. Muitos dos sistemas de gestão de *workflow* disponíveis actualmente disponibilizam funcionalidades para exportar processos para ferramentas de simulação. Esta característica de exportação é utilizada para analisar os aspectos quantitativos dos processos de *workflow*. Um exemplo é a interligação entre o *Staffware* e o *Structware/BusinessSpecs* (IvyTeam, Zug, Switzerland). Outro exemplo é a interligação entre o COSA e o ExSpect (Deloitte & Touche Bakkenist, The Netherlands).

Para ilustrar as funcionalidades destas ferramentas de análise vamos descrever de forma breve dois produtos: o Woflan e o ExSpect.

Woflan. O Woflan (*WO*rk*FL*ow *AN*alyzer) é uma ferramenta que analisa definições de processos de *workflow* especificados em termos de redes de Petri. Foi desenvolvido para verificar definições de processos que são carregados a partir de sistemas de gestão de *workflow* tais como o Staffware e o COSA. Como foi indicado no capítulo 4, há uma necessidade clara para tais ferramentas de verificação.

Hoje em dia, os sistemas de gestão de *workflow* não verificam se a definição de um processo de *workflow* está correcta. Isso pode originar erros durante a fase de modelação pois não podem ser detectados *deadlocks* e *livelocks*. Isto significa que um *workflow* com erros pode avançar para a fase de desenvolvimento, causando problemas dramáticos para a organização. Para

evitar estes problemas de custos elevados, é muito importante verificar a correcta definição dos processos de *workflow* antes de se tornarem operacionais.

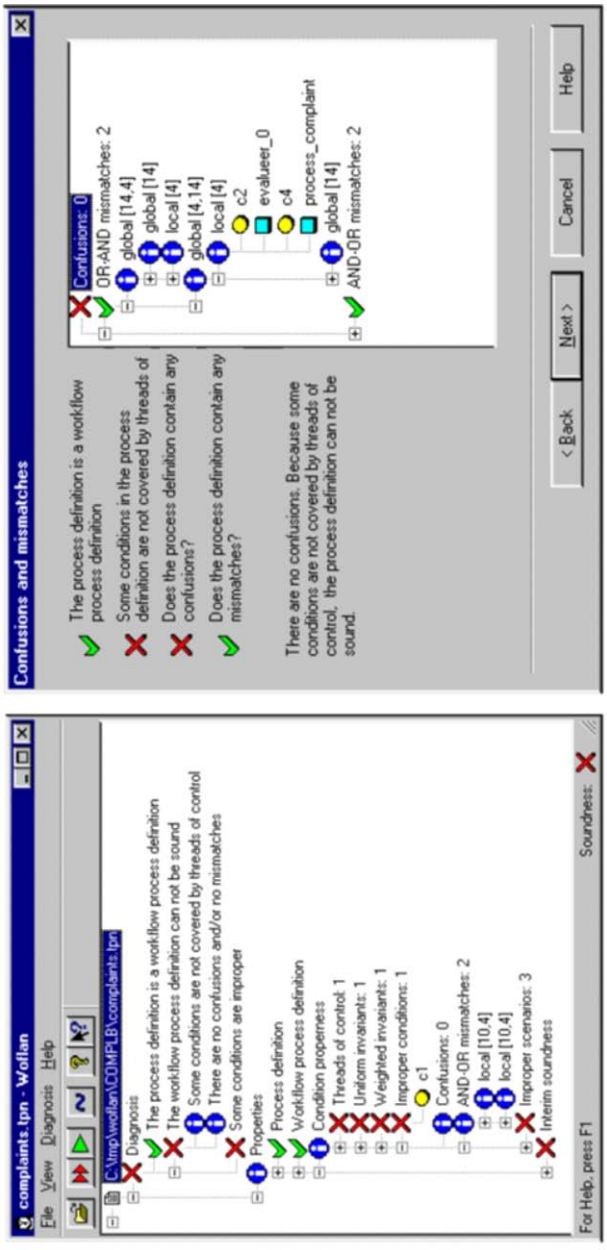


Figura 5.19. O Woflan 2.0 analisando um processo de *workflow* desenhado com o COSA

O desenvolvimento da ferramenta Woflan começou no fim de 1996 e a primeira versão foi lançada em 1997. Basicamente, o Woflan recebe uma definição de processo de *workflow* importado de algum produto de *workflow*, traduz essa definição para uma rede de Petri, e indica se a rede é ou não uma rede de *workflow* correcta. Além disso, usando técnicas de análise de redes de Petri e também técnicas específicas para analisar redes de *workflow*, a ferramenta fornece informação de diagnóstico sobre a rede indicando quando esta não é correcta. A versão 2.0 do Woflan contém uma funcionalidade para importar processos do COSA, do Staffware, do METEOR e do Protos. A figura 5.19 mostra o Woflan. Uma versão experimental do Woflan pode ser obtida em http://www.tm.tue.nl/it/woflan.

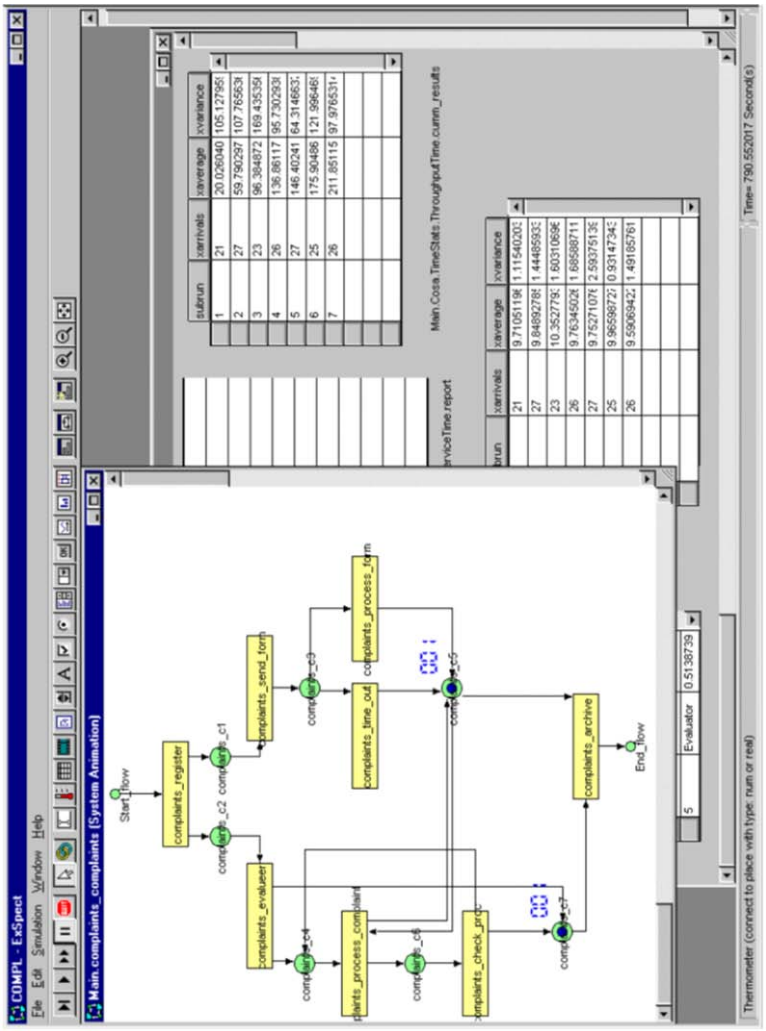


Figura 5.20. O ExSpect simulando um processo de *workflow* desenvolvido usando o COSA

ExSpect. O *ExSpect* (*Executable Simulation Tool*) é uma ferramenta de simulação baseada em redes de Petri. O desenvolvimento do ExSpect começou em 1988 na Eindhoven University of Technology como um protótipo de investigação. Em meados de 1990 o desenvolvimento mudou para uma empresa de consultadoria holandesa, Bakkenist. Actualmente, o ExSpect é suportado pela Deloitte & Touche Bakkenist, The Netherlands. A aplicação ExSpect não está limitada a análises de *workflows*.

O ExSpect pode também ser utilizado na simulação de processos de produção, redes de transporte, componentes de *software*, sistemas embebidos, etc. Na realidade, o ExSpect pode ser utilizado na criação de protótipos de sistemas simples e interagir com sistemas em tempo real via o *standard* COM da Microsoft. Apesar disso, neste livro, a ligação entre o ExSpect e os vários tipos de produtos de *workflow* é muito relevante. O ExSpect pode receber processos de *workflow* de sistemas de gestão de *workflow* como o COSA e ferramentas BPR como o Protos. A figura 5.20 mostra o ExSpect. A figura ilustra que o ExSpect suporta animações gráficas de processos de *workflow*. Além disso, o ExSpect calcula intervalos de confiança para todos os tipos de métricas (tempo, utilização, etc.). É também possível modificar automaticamente modelos de simulação criados a partir do *workflow* para suportar jogos de gestão. Uma versão experimental do ExSpect pode ser obtida em http://www.exspect.com.

5.5.5 Ferramentas de BPR

No capítulo 3 foi explicado que existe um relacionamento próximo entre o Business Process Re-engineering (BPR) e a gestão de *workflows*. Logo, também existem ligações entre as ferramentas de suporte ao BPR e os sistemas de gestão de *workflow*. Algumas das ferramentas que suportam os esforços do BPR centram-se exclusivamente na simulação. O ExSpect é um exemplo desse tipo de ferramenta. Outras ferramentas centram-se na modelação de processos de negócio sem um suporte real para a análise. Exemplos dessas ferramentas que focam exclusivamente na modelação são o Protos (Pallas Athena BV, Plasmolen, The Netherlands) e o ARIS (IDS Scheer AG; Saarbrücken, Germany). Algumas ferramentas oferecem ambas estas funcionalidades, a simulação e as potencialidades de modelação extensivas de processos de negócio, tais como o BusinessSpecs (IvyTeam, Zug, Switzerland), a Income (Promatis AG, Karlsbad, Germany) e o Meta WorkflowAnalyzer (Meta Software, Cambridge, MA, USA). Para ilustrar as funcionalidades destas ferramentas iremos introduzir brevemente o Protos.

Protos. O Protos é uma ferramenta que pode ser utilizada para modelar e documentar processos de negócio. A ferramenta é de fácil utilização e é particularmente útil na modelação de processos de *workflow*, isto é, processos orientados a casos. Apesar do Protos não ser baseado em redes de Petri, consegue suportar as técnicas de esquematização utilizadas neste livro. O Protos suporta a modelação gráfica de processos, documentos, aplicações, papéis, grupos e de equipas. As capacidades de análise do Protos são limitadas: apenas algumas dependências básicas podem ser analisadas (por exemplo, a análise de papéis/encaminhamentos comparáveis às *swimlanes* em UML). O Protos tem

excelentes capacidades para gerar relatórios. É possível gerar automaticamente documentos RTF e páginas HTML com hiperligações. O Protos suporta funcionalidades de exportação para a ferramenta de simulação ExSpect. Existem também interfaces para sistemas de gestão de *workflow* como o COSA (Ley GmbH), o Corsa (BCT), e o FLOWer (Pallas Athena). A figura 5.21 mostra uma imagem do Protos. Para mais informação consulte http://www.pallas-athena.com.

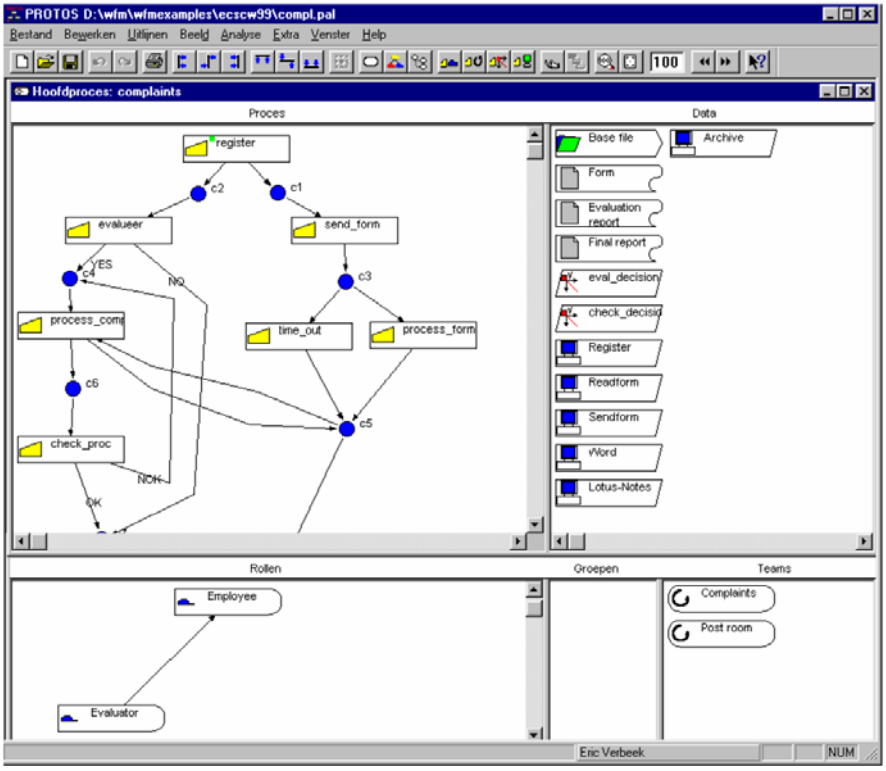


Figura 5.21. Um modelo Protos de um processo de tratamento de reclamações

5.5.6 Escolher um sistema de gestão de workflow

Escolher um sistema de gestão de *workflow* não é uma tarefa fácil. Existem inúmeros aspectos que precisam de ser tomados em consideração. O processo de selecção começa com a criação de uma lista de requisitos que o sistema tem de satisfazer. Com base nesses requisitos, uma *shortlist* é compilada. Durante este processo é necessário ter em consideração as características que são fáceis de verificar, tais como a fiabilidade do fornecedor e se o sistema operativo ou sistema de gestão de base de dados são suportados. Essa *shortlist* deve conter preferencialmente cerca de cinco sistemas.

Cada solução da *shortlist* é, então, sujeita a um exame minucioso. Uma forma para adquirir rapidamente uma boa impressão de um sistema de gestão de

workflow é desenhar *um processo previamente escolhido que representa uma amostra*. A maioria dos fornecedores está disposto a cooperar com um potencial comprador no desenvolvimento do processo. É bastante importante que o processo amostra seja representativo dos processos de negócio. Por exemplo, devemos assegurar que todos os tipos de encaminhamento estão abrangidos. O processo amostra pode ser usado para testar os requisitos funcionais e de desempenho do sistema de *workflow*.

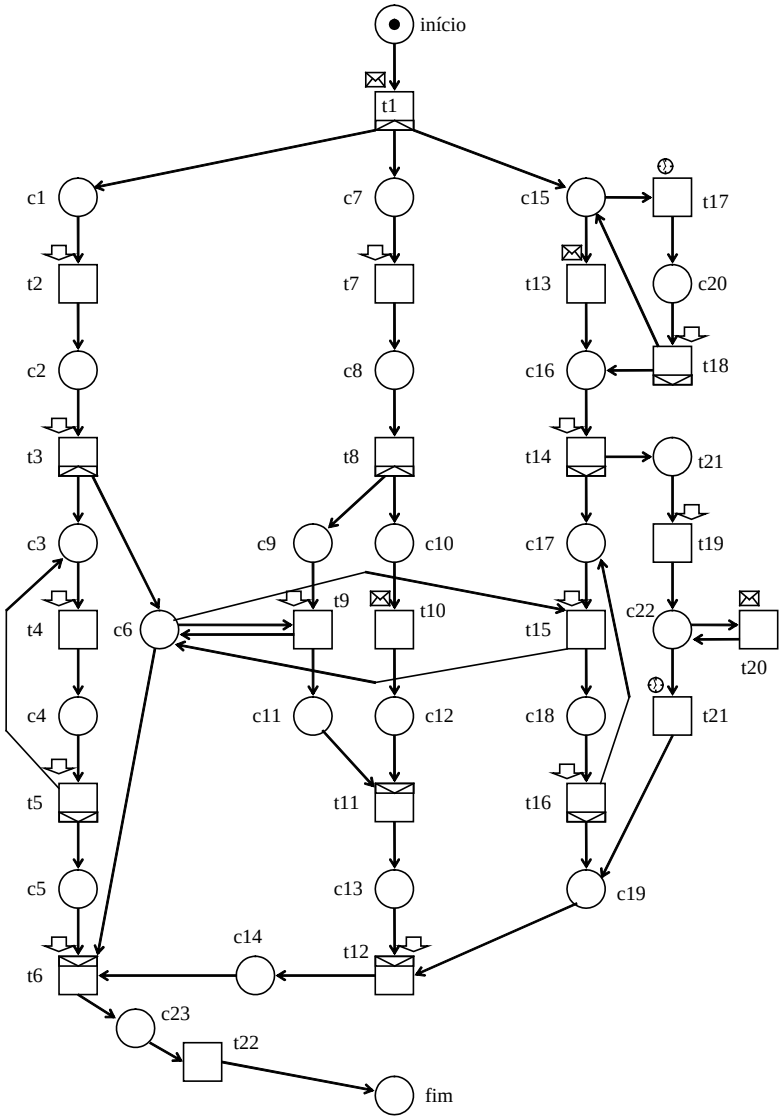


Figura 5.22. Um processo amostra para avaliar um sistema de gestão de *workflow*

A figura 5.22 ilustra um possível processo amostra que, por uma questão de conveniência, iremos chamar *P*. O processo *P* pode ser usado para verificar os requisitos funcionais. Todos os tipos de encaminhamento estão representados, e é usada uma série de diferentes disparos. O processo é relativamente pequeno para estudar o desempenho dos sistemas de gestão de *workflow*. Contudo, se produzirmos um processo em que *P* é representado recursivamente quatro vezes usando sub-processos, criamos algo com maior amplitude.

Comparando o desempenho do sistema quando os quatro sub-processos são executados em paralelo (ligados por um AND-split e um AND-join) e quando existe um encaminhamento selectivo entre eles (os quatro sub-processos são ligados através de um OR-split e de um OR-join), ganhamos uma boa percepção do desempenho do motor de *workflow*. Em ambos os casos, o processo completo consiste em noventa tarefas, isto é suficiente para a maioria das aplicações.

Geralmente, depois de ter testado os sistemas de gestão de *workflow* referenciados na *shortlist*, torna-se claro qual dos pacotes de *software* é a escolha mais acertada.

5.6 Workflow Adaptável

5.6.1 Gestão de workflow e CSCW

De momento, há mais de duzentos produtos de *workflow* disponíveis a nível comercial e muitas organizações estão a introduzir tecnologias de *workflow* para suportar os seus processos de negócio. Reconhece-se que os sistemas de gestão de *workflow* devem fornecer *flexibilidade*. No entanto, os sistemas de gestão de *workflow* actuais têm problemas quando confrontados com *mudanças*. O aparecimento de novas tecnologias, novas leis e novas exigências de mercado podem conduzir a mudanças (estruturais) da definição dos processos de *workflow*. Para além disso, mudanças ad hoc podem ser necessárias devido ao aparecimento de excepções. A incapacidade de lidar com várias mudanças limita a aplicação dos sistemas de gestão de *workflows* actuais.

A figura 5.23 mostra os diferentes campos de suporte ao trabalho colaborativo. Fazemos uma distinção entre aproximações não estruturadas centradas em informação (*Computer-Supported Co-operative Work* ou CSCW) e aproximações estruturadas centradas em processos (*workflows* de produção). As ferramentas existentes estão tipicamente num dos dois extremos do espectro: os produtos de *groupware*, tais como o Lotus Notes e o Exchange, que são ferramentas típicas de CSCW, que não disponibilizam muito suporte aos processos, por outro lado, os WFMSs comerciais disponíveis, tais como o Staffware, o COSA e o MQ Series, não conseguem lidar com construções não estruturadas.

Interligar sistemas de gestão de *workflow* de produção a produtos de *groupware* não resolve o problema, porque a lógica dos processos continua a ser tratada pelo mesmo mecanismo de *workflow* inflexível. Para preencher as lacunas entre os sistemas de CSCW e *workflows* de produção, vários investigadores estão a trabalhar nos problemas associados a *workflows adaptáveis*. Os *workflows* adaptáveis visam fornecer suporte aos processos, tal

como os sistemas de *workflows* normais o fazem, mas de tal forma que o sistema seja capaz de suportar determinadas mudanças. Estas mudanças podem variar entre mudanças ad hoc, tais como a mudança na ordem de duas tarefas para um caso individual (frequentemente chamadas de *excepções*), ao redesenho de um processo de *workflow* como resultado de um projecto de redesenho de processos de negócio (BPR).

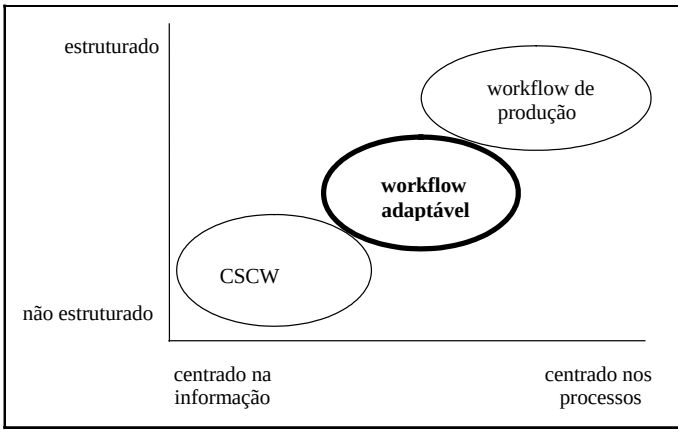


Figura 5.23. O espectro do trabalho colaborativo

Questões típicas relacionadas com *workflows* adaptáveis incluem:

- 1. *Correcção*. Que tipo de mudanças são permitidas e é o resultado da definição de um processo de *workflow* correcta respeitando o critério especificado? Distinguimos correcção sintáctica (e.g., existem nós sem ligações no grafo?) e correcção semântica (e.g., podem os casos existentes no sistema terminarem de forma apropriada?).
- 2. *Alteração dinâmica*. De que forma são tratadas as instâncias em execução (casos) de um *workflow* cuja definição foi alterada? O termo, alteração dinâmica, refere-se aos problemas que ocorrem quando os casos em execução têm de migrar de uma definição de processo para outra.
- 3. *Gestão de informação*. Como fornecer a um gestor informação agregada acerca do estado actual do processo de *workflow*?

Tendo em conta estas questões, apresentamos uma classificação dos tipos de alterações que podem ser realizadas num *workflow*.

5.6.2 Classificação de alterações

Esta secção trata dos diferentes tipos de alterações e das suas consequências. Algumas das perspectivas relevantes para a mudança são:

- 1. *perspectiva de processo*, i.e., as tarefas são adicionadas ou removidas ou a sua ordem é alterada,

- 2. *perspectiva de recurso*, i.e., os recursos são classificados de uma forma diferente ou novas classes são introduzidas,
- 3. *perspectiva de controlo*, i.e., mudar a forma como os recursos são associados aos processos e as tarefas,
- 4. *perspectiva da tarefa*, i.e., fazer um *upgrade* ou um *downgrade* das tarefas,
- 5. *perspectiva do sistema*, i.e., mudanças na infra-estrutura ou na configuração dos motores de execução de casos.

Para sistemas de gestão de *workflow*, a perspectiva dos processos é dominante. Consequentemente, focamos a perspectiva dos processos quando classificamos diferentes tipos de alterações nos *workflows*.

Antes de mais, podemos classificar as alterações com base na abrangência ou no impacto da mudança. Usando este critério, identificamos dois tipos de alterações:

- *Alterações individuais (ad hoc)*. Adaptação ad hoc do processo de *workflow*: um único caso (ou um conjunto limitado de casos) é afectado. Um bom exemplo é o de um hospital: se alguém entrar no hospital com uma paragem cardíaca, o médico não irá pedir-lhe pela sua identificação, embora o processo de *workflow* o indique. Dentro da classe de alterações ad hoc é possível distinguir entre as *entry time changes* (mudanças que ocorrem quando um caso ainda não está no sistema) e *on-the-fly changes* (quanto a definição do processo para um caso, já no sistema, muda).
- *Alterações estruturais (evolutivas)*. Evolução do processo de *workflow*: todos os novos casos beneficiam da adaptação. Uma alteração estrutural é tipicamente o resultado de um esforço BPR. Um exemplo de tal alteração é a mudança de um curriculum de quatro anos, numa universidade, para um de cinco anos.

Existem três maneiras diferentes de alterar um *workflow*:

- 1. a definição do processo é *estendida* (ex., adicionando novas tarefas para suportar extensões do processo),
- 2. as tarefas são *substituídas* por outras tarefas (ex., a tarefa é refinada num sub-processo) e
- 3. as tarefas no processo são *reordenadas* (ex., duas tarefas sequenciais são colocadas em paralelo).

Se uma mudança ocorrer, pode afectar os casos em execução. Manipular casos existentes no sistema quando a definição de um processo muda, cria potenciais problemas. Tratar dos casos existentes só é relevante quando se dá uma alteração estrutural, visto que as alterações individuais são sempre (similares a) excepções e tais são tratadas por quem explicitamente as iniciou. Para as alterações estruturais existem três alternativas: (a) *reiniciar*: os casos em execução são rebobinados (*rolled back*) e reiniciados no início do novo processo, (b) *prosseguir*: alterações que não afectam os casos em execução por ser possível ter várias versões do processo, e (c) *transferência*: um caso é transferido para o novo processo. O termo alteração dinâmica é usado para referir esta última política.

5.6.3 InConcert

Actualmente, muitos investigadores estão a trabalhar em problemas relacionados com *workflows* adaptáveis. Poucos sistemas comerciais fornecem suporte para este tipo de *workflows*. Os problemas relacionados com alterações dinâmicas são difíceis de resolver e não são resolvidos pelos actuais sistemas. Apenas para alterações individuais é que existem alguns sistemas disponíveis. Estes sistemas são denominados por sistemas de *workflow ad hoc*. Nesta secção descrevemos um destes sistemas.

InConcert (TIBCO Software Inc.) é um sistema de gestão de *workflows* desenhado para desenvolver *workflows* flexíveis. A ferramenta tem duas funcionalidades únicas. Em primeiro lugar, o sistema suporta o "desenho de *workflows* por descoberta." Esta funcionalidade permite a criação de *templates* baseados na actual execução de tarefas de *workflow* para um dado caso. Em segundo lugar, o InConcert suporta uma noção de hierarquias de classes que permitem que um objecto InConcert herde funcionalidades de um outro objecto InConcert; por outras palavras, os atributos de uma definição de processo de *workflow* pai podem ser herdados pela definição de um processo de *workflow* filho.

Usando a aplicação cliente InConcert é possível disponibilizar as seguintes ferramentas:

- 1. *Process Designer*. É a ferramenta usada para desenhar definições de processos de *workflow*. Esta ferramenta também pode ser usada para alterar definições de processos de *workflow on-the-fly*.

- 2. *Task User Interface Designer*. É usada para desenhar a interface gráfica apresentada aos utilizadores quando estes estão a executar tarefas.

- 3. *Work Group Manager*. É usado para definir novos grupos de trabalho e para controlar a carga de trabalho dos grupos.

- 4. *Process Manager*. É usado para iniciar e gerir casos (instâncias de *workflows*).

- 5. *Document Organizer*. É usado para organizar e criar documentos InConcert.

- 6. *Task Organizer*. É usado para exibir e executar itens de trabalho.

A figura 5.24 mostra a ferramenta Process Designer do InConcert. A linguagem de modelação utilizada pelo InConcert corresponde a uma subclasse das redes de Petri: *Acyclic Marked Graphs* (AMG). Esta é a classe das redes de Petri sem ciclos e em que cada lugar não pode ter nem transições múltiplas de entrada, nem transições múltiplas de saída. O InConcert não fornece qualquer OR-split explícito ou OR-join. Cada tarefa é considerada como sendo um AND-split e um AND-join. De modo a permitir o encaminhamento condicional, cada tarefa tem uma condição Booleana associada (chamada de *condição de execução*). Uma condição de execução pode ser usada para transpor tarefas. O

desenho do *workflow* ilustrado na figura 5.24 mostra o processo que gere participações de seguro.

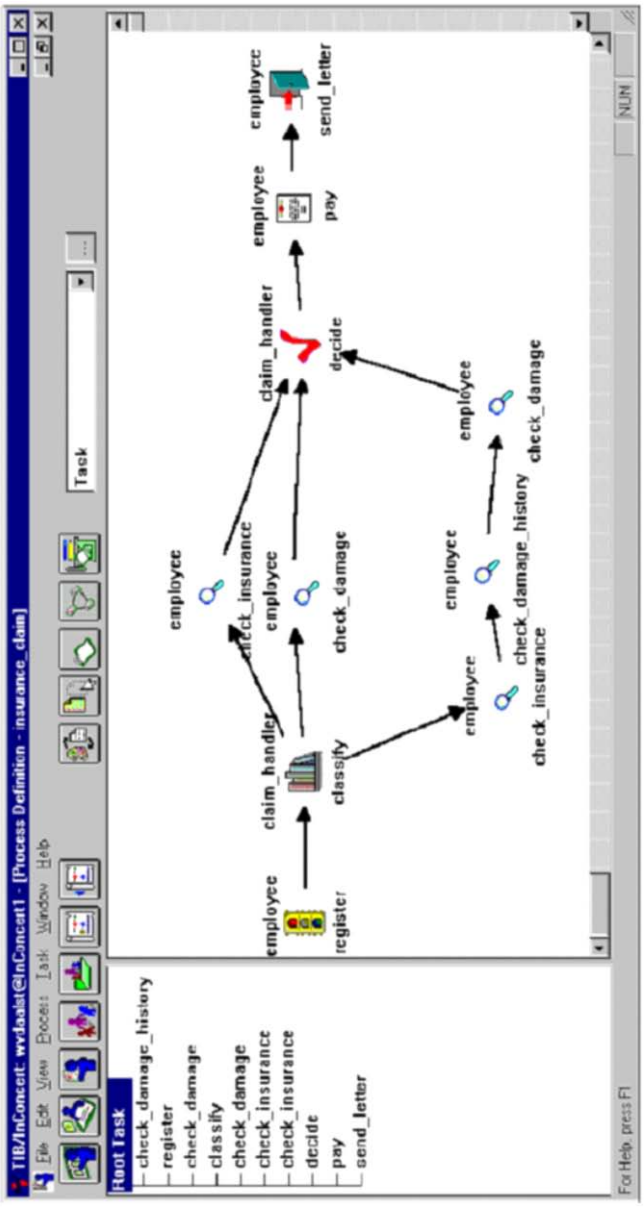


Figura 5.24. A ferramenta Process Definition do InConcert

A tarefa *pay* tem uma condição de execução que indica que só deve ser executada se o resultado da tarefa *decide* for positivo. As tarefas de verificação na figura 5.24 também têm uma condição de execução: ou as duas verificações

paralelas (da parte de cima do processo) ou as três verificações sequenciais (da parte de baixo do processo) são executadas.

O facto do InConcert não permitir OR-splits, OR-joins e iterações, simplifica o processo de modelação. Os designers de *workflows* não podem fazer modelos de *workflow* que entrem em *deadlock* ou que nunca acabem. A definição de um processo de *workflow* é sempre correcta (consultar o capitulo 4). Isto faz com que o InConcert seja um sistema onde os utilizadores possam desenhar ou modificar definições de processos. Ao contrário dos sistemas de produção de *workflow*, o InConcert associa uma definição única de um processo a cada caso individual (i.e., uma instância de *workflow*). Existem várias formas de criar novas instâncias de *workflow*:

1. Instanciar a definição de um processo de *workflow* existente: uma cópia da definição do processo é feita e a primeira tarefa é activada sem alterar o *workflow*.

2. Instanciar uma versão personalizada de uma definição de um processo de *workflow* existente: uma cópia do processo é feita e é alterada de modo a permitir encaminhamentos do tipo *ad hoc*.

3. Instanciar processos de *workflow ad hoc* especificando uma sequência de tarefas e utilizadores.

4. Instanciar um "processo livre de encaminhamento", isto é, um processo de *workflow ad hoc* vazio. Não existe nenhuma definição explícita de um processo de *workflow*: o *workflow* é criado *on-the-fly*.

Instanciar a definição de processo de *workflow* existente corresponde à forma como os casos são tratados nos sistemas de *workflow* de produção tradicionais. A única diferença é que o caso não se refere a uma definição de processo de *workflow* comum, mas a uma cópia privada da definição. Ao criar uma cópia e a possibilidade de alterar essa cópia, tanto na altura da criação como *on-the-fly*, a definição de um processo de *workflow* serve de *template*. Em vez de criar uma cópia do *template*, é também possível criar um processo *ad hoc* a partir do nada. Pelo facto de cada instância de *workflow* possuir as suas próprias definições de processo de *workflow* permite mudanças *on-the-fly*. Em princípio, é possível modificar o encaminhamento de um caso em qualquer altura. Desta forma, as alterações *ad hoc* são totalmente suportadas. Além disso, o InConcert suporta o "desenho de workflows por descoberta". O encaminhamento de qualquer instância de *workflow* completa pode ser usada para criar um novo *template*. Desta forma, as execuções de um *workflow* podem ser usadas para criar definições de processos de *workflow*. A figura 5.25 ilustra o InConcert durante a alteração da definição de processo de uma instância em execução.

O InConcert também suporta o conceito de classe. Existem três tipos de classes: classes de processos, classes de tarefas e classes de documentos. Estas classes estão agrupadas numa hierarquia de classes, onde uma classe filho herda os atributos da classe pai. A classe *Job* é a classe pai de qualquer definição de processo. Quando se define uma classe filho *Activity_based_costing_processes*, todos os atributos são herdados e novos atributos de custo podem ser adicionados. Qualquer definição de processo desta nova classe possui estes

novos atributos. De modo semelhante, é possível definir classes de tarefas e de documentos. O conceito de classe incentiva a reutilização e é uma forma uniforme de realizar o suporte do *workflow*.

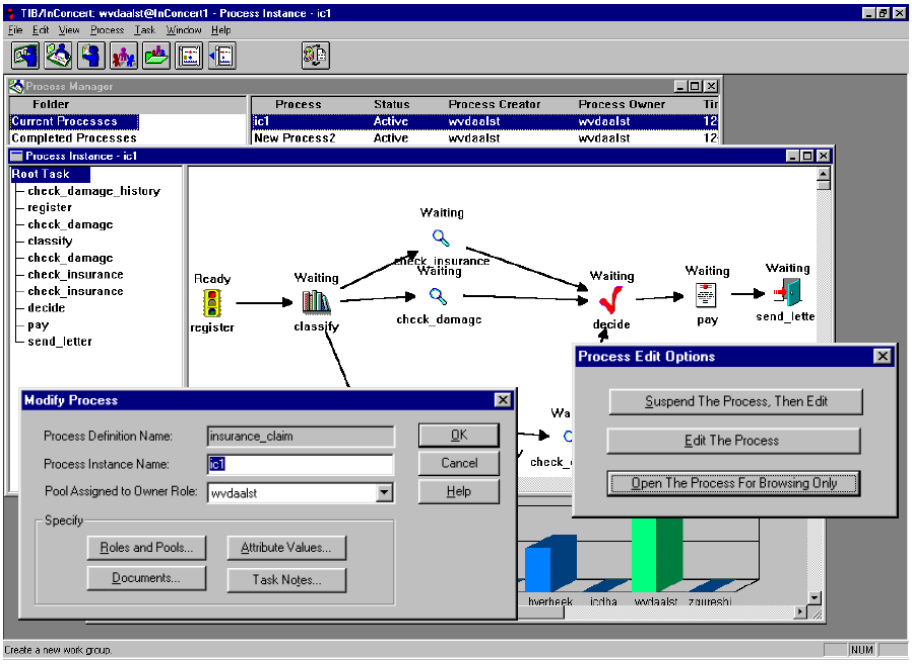


Figura 5.24. Alteração da definição do processo de uma instância *on-the-fly*

5.7 Tendências de Gestão de Workflows

No presente existem muitos fornecedores de sistemas de gestão de *workflow*. Os produtos introduzidos no mercado ainda estão em rápido desenvolvimento. É uma peculiaridade vista em todo o *software* genérico: os fabricantes estão, ao que parece, numa corrida. Cada um tenta incorporar, o mais cedo possível, funções bem sucedidas dos seus concorrentes nas suas novas versões dos produtos, assim como algumas *características próprias e únicas*. Graças a estes desenvolvimentos, podemos ver os pacotes convergir entre si – embora ainda haja diferenças. Está claro que as funcionalidades desejadas pelo WFMC ainda estão longe de ser alcançadas. Ainda não existe experiência prática suficiente para que possamos saber precisamente que funções os sistemas de gestão de *workflows* irão eventualmente abranger. No entanto, é interessante sumariar o seu potencial futuro.

Como veremos, os sistemas de gestão de workflows têm muitas aplicações práticas. Mas isso representa também uma ameaça, uma vez que os fabricantes de outros componentes de *software* genérico – tais como sistemas de gestão de base de dados e sistemas logísticos/pacotes ERP – irão incorporar também as

funcionalidades de gestão de workflows nos seus próprios produtos, eliminando a justificação para a existência de sistemas de gestão de *workflow* separados.

Devemos examinar as perspectivas futuras para os sistemas de gestão de *workflow*, em termos das *oportunidades* e das *ameaças*, em torno de sete áreas de funcionalidade:

1. Modelação;
2. Análise;
3. Planeamento;
4. Gestão de transacções;
5. Interoperabilidade;
6. Internet/Intranet; e
7. Gestão logística.

Porque o software específico para cada uma das áreas acima enumeradas está também disponível, devemos considerar as ameaças ao lado das oportunidades (isto é, as possibilidades de aplicação).

5.7.1 Modelação

Uma das funções mais importantes dos sistemas de gestão de *workflow* é a modelação de workflows. Esta habilidade significa que um sistema pode ser considerado um *repositório* para *dados de metabusiness*: a informação estrutural de uma organização, tal como os seus processos e diagramas organizacionais. A estas ferramentas foi dado o nome de *orgware* (advém de "organization-ware"). No entanto, há repositórios específicos em que podem ser guardados mais do que os dados de uma organização: por exemplo, todos os tipos de *indicadores de desempenho* de processos de negócio, um *modelo de dados da organização* ("um dicionário de dados" de todas as bases de dados que usa) e um *mapa* dos seus sistemas de informação. A vantagem de tais repositórios é que oferecem boas oportunidades de consulta com as quais todas as ligações relevantes à gestão da organização podem ser analisadas. Os repósitos são frequentemente desenvolvidos usando um sistema de gestão de base de dados e/ou uma ferramenta OLAP (on-line analytical processing). Uma diferença essencial entre estes é que as ferramentas OLAP permitem que estruturas hierárquicas possam ser pesquisadas recursivamente (conhecido como "downdrilling"), o que não é possível em SQL (a linguagem de consulta usada em sistemas de gestão da base de dados relacionais). Consequentemente, é óbvio que os sistemas de gestão de *workflow* irão adquirir mais funções no futuro, ou irão melhorar a interface com essas ferramentas.

Outro aspecto importante é o *poder expressivo* da função de modelação na actual geração de sistemas de gestão de *workflow*. Muitos dos sistemas existentes não têm um bom modelo para representar os processos. Isto significa que determinadas construções comuns em processos de negócio não são tratadas como deve ser. Este problema será certamente resolvido, e pode-se esperar que eventualmente todos os sistemas de gestão de *workflow* modelam os seus processos de uma forma semelhante à teoria das redes de Petri.

Um aspecto final sobre a modelação é que os sistemas de gestão de *workflow* de hoje em dia estão principalmente ajustados para processos *standard*. Por

outras palavras, a ferramenta de definição de um processo descreve um número de processos de negócio pelos quais muitos casos são executados. Geralmente, visto que o número de casos é relativamente grande quando comparado com o número de processos, designamos a estes processos de *workflows de produção*. No futuro, no entanto, devemos também esperar sistemas que ofereçam funcionalidades para processos *únicos* (*workflows ad hoc*), com um processo definido separadamente para cada caso.

Uma complicação adicional pode ocorrer quando uma alteração é empreendida num processo quando um caso estiver a ser processado. Encontramos exemplos destes na indústria de transportes (quando decisões para alterar as rotas são feitas na estrada) e nos serviços de saúde (quando o tratamento apropriado só pode ser decidido após a fase do diagnóstico). Nos sistemas de gestão de *workflow* actuais isto pode ser parcialmente superado definindo um processo com várias tarefas genéricas (mas esta solução desloca apenas o problema). O uso de tarefas genéricas implica que uma parte considerável da gestão tenha de ser feita dentro das aplicações. Resolver o problema significará, posteriormente, uma maior integração entre a funcionalidade da definição dos processos com os motores de *workflow*.

5.7.2 Análise

Os novos processos de negócio têm de ser analisados a fim de se estabelecer se serão bem executados no sentido quantitativo (tempos de conclusão, utilização dos recursos, e assim por diante) e qualitativo (se estão correctos, se funcionam bem e se podem ser executados pelos funcionários da organização). Quando os processos existentes são melhorados, a análise dos processos modificados é também desejável antes que as mudanças sejam postas em prática. Para realizar uma análise, podemos usar a simulação e diversas técnicas formais da verificação. O alargamento destas capacidades é um óbvio e necessário desenvolvimento futuro. Para a simulação, isto significa que será mais fácil usar dados históricos dos sistemas de gestão de *workflow* para testar processos de negócio modificados e permitirá desenhar cenários hipotéticos que poderão potencialmente conduzir a novas mudanças. Por outras palavras, os funcionários que desempenham uma parte dos processos procuram fraquezas no sistema de gestão de *workflow* usando um jogo de simulação de negócio. Esta função também pode ser usada para treinar novos funcionários. Diversos sistemas de gestão de *workflow* existentes oferecem já algumas facilidades de jogo, mas há muito espaço para melhoramento – por exemplo, suportar o *rollback*.

Existem muitas ferramentas de simulação no mercado e não é impensável que estas se desenvolvam na direcção dos sistemas de gestão de *workflow*. Apesar de tudo, não existe uma diferença assim tão grande entre simular workflows e coordenar workflows reais. Consequentemente, é possível que algumas ferramentas de simulação possam evoluir para motores de *workflow*. Assim como a simulação, existem também métodos formais de análise, que deixam muito a desejar. Os métodos disponíveis foram desenvolvidos principalmente para as redes de Petri e não são particulares para estruturas específicas de processos de negócio. É muito provável que diversos testes de correcção, como os oferecidos pelo sistema Woflan, sejam incorporados nas

ferramentas de definição de processos no futuro. Se o desenhador cometer um erro, isto vai "dificultar-lhe a vida", a não ser que compreenda a teoria subjacente aos testes realizados.

5.7.3 Planeamento

A geração actual dos sistemas de gestão de *workflow*, às vezes, oferece apenas uma capacidade limitada de atribuir recursos às tarefas e de decidir a ordem em que as tarefas que usam os mesmos recursos devem ser executadas (este tipo de planeamento é conhecido por *scheduling*). Os sistemas existentes, virtualmente, não prestam atenção aos *problemas de escalonamento* que ocorrem ao organizar os recursos humanos. E, devido ao aumento da flexibilidade laboral e ao alongamento das horas de negócio das organizações, este problema está a tornar-se cada vez mais significativo. A funcionalidade é requerida, mas, presentemente, não se encontra suficientemente suportada pelos sistemas de gestão de *workflow*.

Um melhor suporte no planeamento deverá ser oferecido por aplicações modernas de investigação operacional e de inteligência artificial na preparação de listas e planos. Tais métodos, como o *simulated annealing*, *taboo search* e *constraint satisfaction*, foram reconhecidos na prática. Ao lado destes problemas *operacionais* de planeamento existem também problemas *tácticos*. Estes estão relacionados com as decisões sobre a *capacidade* de recursos específicos (não apenas os humanos) que são requeridos durante o período de planeamento. Embora um sistema de gestão de *workflow* contenha de facto toda a informação relevante necessária para resolver tais problemas, na realidade nenhum oferece funcionalidades para o fazer. Outra questão é se os produtores destes sistemas devem desenvolver tais funcionalidades, ou se seria melhor tentar integrar *software* proprietário de planeamento nos seus programas.

5.7.4 Gestão de transacções

A grande maioria dos sistemas de gestão de *workflow* limitou-se a gerir processos dentro de uma única organização. Ao fazer isso, assumem que os recursos (humanos) são empregados exclusivamente por essa organização e podem ser distribuídos à vontade pelo gestor de recursos (o chefe ou o sistema de gestão de *workflow*). Consequentemente, assume-se que todos os recursos humanos têm o mesmo *software* cliente e que toda a troca de informação ocorre de uma forma uniforme. Se desejarmos aplicar sistemas de gestão de *workflow* para coordenar processos de negócio em empresas virtuais ou em organizações integradas em rede, então surgem vários problemas que não podem ser tratados pelos sistemas actuais. Note-se que os sistemas de gestão de *workflow* são muito relevantes no suporte do processamento de transacções *e-business*. No entanto, eles necessitam de funcionalidades adicionais para suportar processos inter-organizacionais.

Como foi descrito no capítulo 1, usando um exemplo da indústria de transportes, encontrar um recurso apropriado requer um *processo de comunicações*. Fazendo isso, uma árvore de transacção é passada até que um actor, que esteja disposto a fazer de recurso, seja encontrado. Uma complicação

adicional é que já não podemos assumir que todos os recursos podem interpretar a mesma informação. Os padrões e *standards* de comunicação e conversão, tal como os geralmente usados no mundo EDI, tornar-se-ão, assim, vitais para os *workflow* inter-organizacionais. O XML oferece um *standard* muito prometedor para esse fim. O processo de comunicações entre as partes envolvidas não irá apenas cobrir o tempo dentro do qual a tarefa pode ser terminada, mas também a quantidade de dinheiro associada a ela. Assim, os sistemas de gestão de *workflow* terão também de fornecer funcionalidades para o estabelecimento financeiro do trabalho executado pelos recursos.

Uma complicação interessante da gestão de *workflow* dentro de organizações integradas em rede é que o termo "tarefa" muda. Não é uma parte atómica de trabalho para todos. O que é uma tarefa para o supervisor é uma definição de processo para o fornecedor. É por isto que é vantajoso usar redes de Petri hierárquicas, porque estas podem modelar tais situações facilmente. Se as árvores da transacção (ver o capítulo 1), para encontrar actores apropriados para executar um caso, se tornarem muito altas e cada actor apenas oferecer um compromisso ascendente (uma confirmação da ordem ao seu fornecedor), assim que este tenha obtido tal compromisso dos seus sub-fornecedores, então a aceitação de uma ordem ao nível mais elevado pode transformar-se num negócio extremamente demoroso. Isto forma um "limite natural" para a eficácia das organizações integradas em rede. Nalgumas situações, elas serão apenas praticáveis se o processo de comunicações puder ser em grande parte automático. Tal como com os *standards* de mensagens, os acordos detalhados entre os actores são também requeridos para que isto seja conseguido. Além disso, as funções adicionais para os sistemas de gestão de *workflow* em organizações integradas em rede também terão as vantagens das estruturas hierárquicas. Isso porque, fornecem uma oportunidade para o controlo descentralizado e assim concedem poderes adicionais aos funcionários.

5.7.5 Interoperabilidade

Uma das propriedades interessantes de um sistema de gestão de *workflow* é que os recursos humanos e as aplicações de computador são tratados uniformemente. O sistema organiza todo o trabalho que necessita ser realizado num caso. Ou seja, trata do escalonamento dos recursos e assegura que tenham a informação correcta quando começarem a executar tarefas. Resumindo, o sistema de gestão de *workflow* fornece a gestão logística do trabalho, assemelhando-se assim a um sistema operativo. Da mesma forma, o sistema operativo também executa tarefas para as várias transacções dos utilizadores e trabalho de grupo. A diferença é que um sistema de gestão de *workflow* também controla o trabalho dos recursos humanos que estão fora do sistema.

Um sistema de gestão de *workflow* pode ser, assim, considerado como um tipo de sistema operativo para uma organização. Na teoria, poderia também ser usado para ligar várias aplicações, desde que a ordem das tarefas seja descrita pelo processo de trabalho através de um gráfico de fluxo ou de outra ferramenta semelhante. Assim, tal sistema poderia executar o fluxo de controlo de um grande sistema de informação, no qual os programas realizariam as transformações dos dados. No entanto, embora em princípio seja possível, a geração actual dos sistemas de gestão de *workflow* ainda não fornece este tipo

de uso. Primeiro, as Interfaces de Programação de Aplicações (APIs) *standards* são demasiado limitadas. Em segundo lugar, o sistema de gestão de *workflow* teria de ser capaz de funcionar como um tipo de *software de barramento* entre as várias aplicações – um papel para o qual o seu desempenho ainda é muito inadequado. Também teria de ser possível monitorizar protocolos entre aplicações comunicantes e suportar a conversão de dados entre elas. Além disso, habitualmente não há nenhuma funcionalidade para realizar o *rollback* de transacções e lidar com falhas de *hardware*. Se estas limitações pudessem ser superadas, um sistema de gestão de *workflow* transformar-se-ia numa ferramenta ideal para resolver problemas de interoperabilidade.

5.7.6 Internet/Intranet

Um número limitado de sistemas de gestão de *workflow* permite o uso de um *browser* tal como o Netscape Navigator ou o Microsoft Internet Explorer como aplicação cliente de *workflow* (Interface 2). Nesses casos, não é usado um gestor de listas de trabalho específico, em vez disso, o browser age como o gestor de listas de trabalho. Isto faz com que seja possível acedermos ao sistema de *workflow* através da *Internet*, também conhecida como *World Wide Web* (WWW). Esta aproximação tem um número significativo de vantagens. Primeiro, os utilizadores não estão limitados a um local de trabalho específico. Se o sistema de gestão de *workflow* está ligado a WWW, então em princípio é possível trabalhar em qualquer lugar. Mesmo da Austrália, por exemplo, não há nenhum problema em aceder a um sistema de *workflow* na Europa.

Outra vantagem importante é o facto de podermos utilizar padrões e *standards* amplamente aceites, tais como o HTTP (HyperText Transfer Protocol), o HTML (HyperText Markup Language), o XML (eXtendible Markup Language) e o CGI (Common Gateway Interface). Consequentemente, não existe nenhuma dependência dos protocolos de comunicação especialmente desenvolvidos para um sistema de gestão de *workflow*. O uso de páginas XML/HTML é suficiente. A combinação de workflows e da World Wide Web abre novas oportunidades para aplicações *e-business*. Muitos serviços oferecidos na Web podem ser suportados por um sistema de gestão de *workflow*. Considere por exemplo o processamento de ordens de compra, reclamações, formulários e assim por diante. Curiosamente, estas aplicações tornam confusa a distinção entre o cliente e o funcionário: ambos acedem ao sistema de *workflow* da mesma forma. No entanto, também há alguns problemas associados ao uso da World Wide Web com uma aplicação cliente de *workflow*. Primeiro, a sua velocidade pode deixar muito a desejar, frequentemente, leva algum tempo até que uma tarefa possa ser aberta ou fechada. Nem a segurança é perfeita, pois é difícil de proteger informação confidencial. Estes problemas podem ser resolvidos de alguma forma usando uma *Intranet*. Esta tem a mesma estrutura que a World Wide Web, mas é limitada em extensão. Consequentemente, uma empresa pode "proteger" a sua rede do mundo exterior e as velocidades não são limitados pelo "engarrafamento" como na World Wide Web. Além disso, permite usar os *standards* e os produtos mencionados anteriormente.

Um problema que não pode ser resolvido com uma *Intranet* é o uso massivo das aplicações. Aplicações interactivas, tais como processadores de texto, só podem ser iniciadas através de funcionalidades adicionais e de aplicações a nível

dos dados, resultando num elevado congestionamento na rede. Novos ambientes de desenvolvimento (como o Java e o CORBA) apenas resolvem parcialmente estes problemas. Assim, permanece pouco claro que perspectivas a World Wide Web pode oferecer às gerações futuras dos sistemas de gestão de *workflow*.

5.7.7 Gestão logística

Uma das mais bem sucedidas categorias de aplicações genéricas é a dos sistemas de gestão logística, também conhecidos por *sistemas ERP* (*Enterprise Resource Planning Systems*). Alguns destes pacotes evoluíram a partir de *software* financeiro e desenvolveram-se através da extensão das funcionalidades de gestão de *stocks*. Estes pacotes permitem o suporte de um grande número de funções de negócio de produção (por exemplo, a indústria automóvel, distribuição, transporte, sector bancário, seguros e governo). Uma das suas funções mais importantes é o cálculo dos materiais necessários com base no tempo de produção planeado de um produto. Reciprocamente, os materiais necessários são usados para gerar um plano de escalonamento detalhado. A base para isto é uma *lista de componentes* de produtos, também conhecida por "bill of material" (BOM). Se um produto deve estar pronto numa determinada data e é sabido quanto tempo leva a juntar os seus subconjuntos maiores e a terminar o produto (por exemplo, a pintura), então podemos calcular quando é que os subconjuntos devem estar prontos. Se estes também forem feitos dentro da organização, um plano similar pode ser redigido para os subconjuntos. Se forem comprados externamente, pode ser determinado o prazo de entrega.

A geração actual do *software* de gestão logística não usa o termo "processo de negócio" de modo tão genérico e flexível como os sistemas de gestão de *workflow* o fazem. Naturalmente, os seus vendedores seguem os desenvolvimentos nos sistemas de gestão de *workflow* de perto e são capazes de incorporar algumas das suas funções nas novas gerações dos seus produtos. Dada a estrutura dos seus produtos (*legacy*) é difícil de prever se eles são capazes de o fazer eficazmente. Certamente tais produtos têm muitas outras funções interessantes – particularmente para empresas de produção – e poderiam provavelmente compensar eficazmente o fraco suporte aos workflows.

Em relação a esta ameaça, o revés da medalha representa uma oportunidade que passa pela incorporação de um número de funções de pacotes de logística nos sistemas de gestão de *workflow*. A BOM é de particular interesse. Os sistemas de gestão de *workflow* são sempre baseados num processo composto por um número de tarefas. O conteúdo preciso destas tarefas é inteiramente ignorado, assim como a informação requerida para executá-las. Redigir uma BOM para cada tipo de caso, mostrando que informação é requerida para o terminar, na teoria, permitiria *deduzir* o que são as tarefas. Podemos ilustrar o uso de tal lista usando o exemplo da declaração de seguro do capítulo 1. O caso pode ser fechado quando o montante do pagamento é conhecido e quando o possuidor da apólice de seguro concordar com o pagamento (que pode ser zero). A quantidade é requerida, e por isso, o valor da reivindicação tem que ser estabelecido, e também tem de ser verificado se está de acordo com o contrato de seguro (e assim podemos prosseguir). Desta forma, podemos deduzir o processo a partir das necessidades de informação e o formato dos dados

requeridos para cada tarefa. Começando com a lista de componentes, o desenhador do processo pode começar o seu trabalho num nível mais elevado. Esta lista pode também ser útil para o motor de *workflow*, permitindo recolher a informação que necessita, antecipadamente, e submete-la ao recurso correspondente no momento apropriado.

Analisamos sete grupos de funções que serão importantes para os sistemas de gestão de *workflow* do futuro. Algumas já estão a ser incorporadas na geração mais recente de sistemas. É improvável, contudo, que os fabricantes venham a incorporar todas estas funcionalidades. Isto não seria sensato, porque eles nunca seriam capazes de permanecer actualizados em cada um destes campos. Uma melhor solução é deixar a arquitectura dos seus sistemas suficientemente aberta de modo a que seja fácil integrar pacotes de *software* de outros fabricantes – com funções específicas, tal como foi descrito anteriormente. Mas para tal é necessário muito trabalho na criação de *standards*.

EXERCÍCIOS

Exercício 5.1

Descreva o modelo de referência do WFMC, isto é, forneça um modelo gráfico dos componentes e das interfaces. Descreva cada componente em detalhe. Discuta também a funcionalidade de cada uma das cinco interfaces.

Exercício 5.2

Responda às seguintes perguntas:

(a) O que são as propriedades ACID?
(b) Que interface causa tipicamente problemas técnicos?
(c) Quais são os quatro papéis dos funcionários envolvidos no desenho e na implementação de um sistema de gestão de *workflow*?
(d) Indique alguns exemplos dos *standards* de interoperabilidade para workflows que tratem de aspectos de tempo de execução (*run-time*).
(e) Caracterize os seguintes sistemas de gestão de *workflow*: Staffware, COSA e ActionWorkflow.
(f) Qual é a funcionalidade das ferramentas de análise tais como Woflan e o ExSpect?
(g) Indique algumas ferramentas de BPR.

Exercício 5.3

Modele o processo ilustrado na figura 5.26 usando as linguagens de modelação suportadas pelo Staffware e pelo COSA.

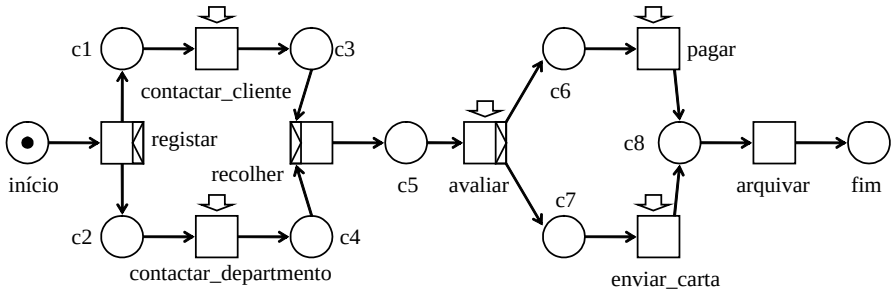


Figura 5.26. Processo "tratar reclamação"

Exercício 5.4

Modele a agência de viagens descrita no capítulo 2 usando as linguagens de modelação suportadas pelo Staffware e pelo COSA.

(Página deixada propositadamente em branco)

Capítulo 6.

DESENVOLVIMENTO DE SISTEMAS DE WORKFLOW

6.1 Métodos de Desenvolvimento

Os capítulos anteriores definiram o que são *workflows*, como é possível modelá-los, e de que formas os sistemas de gestão do *workflow* podem desempenhar um importante papel na execução e na gestão dos processos de negócio.

Neste capítulo descrevemos um método de desenvolvimento específico ou um guia geral para o desenvolvimento de sistemas baseados em software de gestão de *workflow*. Um guia geral é um plano para o desenvolvimento de sistemas. Descreve uma sequência de fases e para cada fase determina quais as actividades a serem efectuadas e quais os produtos finais. O guia indica o *que* deve ser feito, mas não descreve *como* deve ser feito. Consequentemente, um guia é usado em combinação com outros métodos específicos para cada actividade. Para o processo de modelação usamos os métodos apresentados nos capítulos anteriores. Chamamos a este método IPSD, que é um acrónimo para *Interactive, Process-oriented System Development*.

6.1.1 Porquê usar um método específico para a gestão de *workflow*?

Naturalmente, vários métodos de desenvolvimento existentes e já testados podem ser usados para construir sistemas de suporte ao *workflow*. Porquê, então, é necessário um método específico para o desenvolvimento de sistemas de *workflow*?

Os métodos existentes para o desenvolvimento de sistemas de informação atribuem uma maior ênfase na definição de estruturas de dados e na forma como a aplicação é apresentada aos utilizadores (i.e., a interface com o utilizador). Nestes métodos, as mudanças organizacionais e o (re)desenho dos processos recebem uma atenção limitada. O desenvolvimento de uma nova geração de sistemas de *workflow* vai, geralmente, ao encontro de uma reorganização radical dos processos de negócio. Além disso, as oportunidades que o software de gestão de *workflows* fornece para organizar e gerir os fluxos tem consequências que se estendem às relações dentro da organização e na forma como as pessoas colaboram entre si. Um método para desenvolver um sistema de *workflow* deve, consequentemente, centra-se nos processos de negócio e envolver tanto a organização como a tecnologia.

A forma como o processo de desenvolvimento é executado deve envolver os "utilizadores" no desenho de processos e de sistemas, tanto quanto possível. O processo de desenvolvimento deve, preferivelmente, ser evolutivo. Isto significa que as funcionalidades do sistema são melhoradas, com avaliações e revisões contínuas das aplicações de teste ou protótipos, até se provarem satisfatórias. Usando instrumentos de software modernos, tais como as ferramentas CASE e os geradores de software, protótipos iniciais podem ser produzidos com base em especificações genéricas. Estes podem então ser constantemente reajustados com base na experiência e no *feedback* dos utilizadores. O software configurável, tal como os sistemas de *workflows*, também permite este tipo de prototipagem.

O facto que estarmos a descrever um novo método não significa que desejemos "reinventar a roda" a partir do nada. Como base usaremos ideias estabelecidas, tais como o *Business Process Re-engineering* (BPR) e o *Rapid Application Development* (RAD). A integração de técnicas RAD dentro do ciclo BPR fornece um excelente contexto para o desenvolvimento dos sistemas de *workflow*, nos quais o desenvolvimento de processos de trabalho e os sistemas de suporte são completamente integrados. Uma aproximação evolucionária suportada por ferramentas modernas para permitir a prototipagem e a experimentação é um elemento essencial neste esforço de desenvolvimento.

6.1.2 Reengenharia de processos de negócio

Depois de várias décadas de uso de sistemas computacionais, muitas organizações chegaram à conclusão que é necessário adoptar novas aplicações para se conseguirem melhorias adicionais. Muitos sistemas de informação são ainda baseados em métodos de trabalho que remontam à época em que se "escrevia com uma pena". Uma aproximação radical é, consequentemente, necessária para obter um maior rendimento das tecnologias de informação.

O BPR permite, em suma, ser descrito como um esforço para conseguir uma estrutura mais eficaz e eficiente para os processos de negócio. Um factor importante do BPR indica que o ponto de partida para o redesenho não pode ter por base os "processos antigos". A informação e as tecnologias de comunicação são as condições mais importantes para conseguir este objectivo (ver o capítulo 3 para uma definição mais detalhada de BPR).

O BPR segue um ciclo mais ou mais menos fixo, denominado de *ciclo de vida* do BPR. Isto é ilustrado através do diagrama da figura 6.1. O ciclo começa com uma iniciativa, na maior parte dos casos da responsabilidade de um gestor sénior.

O ciclo de vida do BPR tem quatro fases.

1. O ciclo de vida começa com a fase de *diagnóstico*. Esta começa com a análise da situação actual, e, em particular, dos problemas causados pela forma de trabalho corrente. Com base nessa informação, os objectivos têm de ser estabelecidos para que o sucesso e as melhorias possam, posteriormente, ser quantificadas. Os processos existentes são analisados e é feito um diagnóstico onde são identificados os problemas. Entre outras coisas, isto permite mostrar quais os métodos de trabalho existentes que não produzem os resultados esperados.

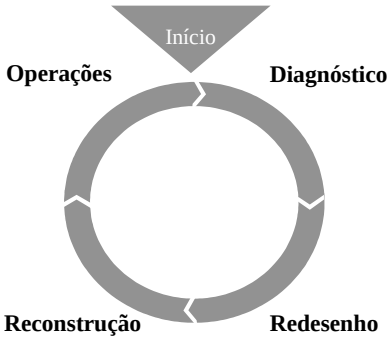


Figura 6.1. O ciclo de vida do BPR

2. Uma vez feito o diagnóstico, inicia-se a *fase de redesenho*. O novo desenho começa usando uma "folha de papel em branco". Por outras palavras, as formas existentes de trabalho não são consideradas. Em vez disso, uma descrição inteiramente nova do processo é produzida – independentemente de limitações como a estrutura organizacional e os recursos disponíveis, e é determinada unicamente pela especificação de *inputs* e *outputs*.

3. A fase de redesenho é seguida da *fase de reconstrução*. Durante esta fase, um novo conjunto de definições de processos, sistemas SI/TI e a estrutura da organização, é criado para suportar os processos previamente identificados.

4. Durante a *fase operacional*, o desempenho dos processos é medido e avaliado usando critérios predefinidos de desempenho. Através destes, potenciais congestionamentos podem ser identificados de forma rápida. Estes podem facilmente justificar o lançamento de um novo ciclo de reengenharia, envolvendo muito possivelmente modificações de uma natureza menos radical do que aquelas que foram feitas nas fases iniciais.

Estes quatro pontos fornecem uma visão geral das actividades envolvidas num projecto de BPR. As actividades cruciais são aquelas que acontecem durante as fases de redesenho e reconstrução. Na secção 6.2 encontraremos estas mesmas fases cujas actividades serão discutidas em mais detalhe.

6.1.3 Desenvolvimento rápido de aplicações

O desenvolvimento rápido de aplicações (RAD) é um método para o desenvolvimento de sistemas que é caracterizado por um processo de desenvolvimento cíclico, uma colaboração estreita com os utilizadores e o uso de ferramentas modernas de desenvolvimento rápido. Os principais objectivos são: rapidez, redução de custo e melhoria da qualidade, graças a um elevado grau de participação do utilizador. Neste livro, baseamo-nos numa aproximação do método rápido de desenvolvimento de aplicações introduzido por James Martin em 1991.

Em termos gerais, o faseamento usado no RAD corresponde às abordagens usadas em métodos mais tradicionais. A diferença centra-se, não na sequência

de actividades, mas na forma como estas são realizadas durante cada fase. Antes de examinarmos as fases do RAD e a metodologia, olhemos primeiro para um conjunto de termos e de técnicas que são cruciais para este.

O RAD é um *processo de desenvolvimento iterativo ou baseado em ciclos*. Por outras palavras, a análise, o desenho e as fases de construção são executadas repetidamente, em pequenas etapas que se sucedem rapidamente. Cada ciclo resulta num produto tangível que é usado como base para começar a fase seguinte. O novo conhecimento adquirido pode constituir de imediato uma contribuição através da actualização do desenho, beneficiando assim a qualidade e a aceitação do sistema. A *prototipagem* é um instrumento importante para estabelecer uma comunicação eficiente e efectiva com os utilizadores. As especificações (de uma parte) do sistema, ou de componentes individuais, são avaliadas usando os protótipos desenvolvidos. Chamamos de *desenvolvimento evolucionário* quando a aplicação de tal método, através do desenvolvimento de protótipos e de uma melhoria gradual, resulta numa aplicação final. As especificações e o sistema "evoluem" simultaneamente num sistema operacional.

Um sistema é frequentemente demasiado grande para ser avaliado na sua totalidade pelo utilizador, e o seu desenvolvimento e execução numa só iteração envolve demasiados riscos. Consequentemente, pode tornar-se útil desenvolver *e* implementar um sistema num número de estágios ou "incrementos" separados. Neste caso, chamamos a este método de *desenvolvimento incremental*. Cada estágio do desenvolvimento termina com a entrega e o lançamento de uma nova versão do sistema que é uma melhoria/expansão do anterior. O desenvolvimento evolucionário e o desenvolvimento incremental são estratégias diferentes, mas que podem ser combinadas eficazmente. Contudo, estes métodos não são o mesmo que a *entrega e execução faseadas*. Este último método é baseado no desenho de um sistema na sua totalidade que é seguido pelas fases de construção, entrega e execução dos módulos da aplicação completa. Isto só é possível quando as secções que estão a ser implementadas não são directamente dependentes de outras partes do sistema que deverão ser entregues mais tarde.

Tais técnicas, como o desenvolvimento evolucionário e a prototipagem, somente podem ser aplicadas com sucesso quando um relacionamento de funcionamento muito próximo pode ser estabelecido entre colaboradores/programadores e utilizadores. Chamamos este relacionamento de *desenvolvimento conjunto* (*joint development*) devido à colaboração próxima e da subsequente responsabilidade colectiva pelo resultado final. Organizar tal colaboração é por si só uma arte. A maioria dos especialistas em tecnologias de informação está habituada ao modelo "parlamentar", no qual os utilizadores só podem submeter alterações às propostas da equipa de desenvolvimento (i.e., ao esboço do relatório final). No desenvolvimento conjunto, os *workshops interactivos* desempenham um papel importante. Em princípio, todos os participantes têm a mesma importância durante estas sessões conjuntas. *Brainstorming*, tomada de decisão, selecção e elaboração são executadas usando técnicas especiais. Porque todos os envolvidos estão presentes e têm um papel activo, as falhas de comunicação são resolvidas e, assim, podem ser tomadas decisões bem fundamentadas. A especificação, prototipagem e testes ocorrem durante estas sessões de trabalho.

A abordagem RAD consiste em quatro fases sucessivas: *planeamento de requisitos*, *desenho centrado nos utilizadores*, *construção* e *entrega*. A figura 6.2 ilustra o relacionamento entre estas fases.

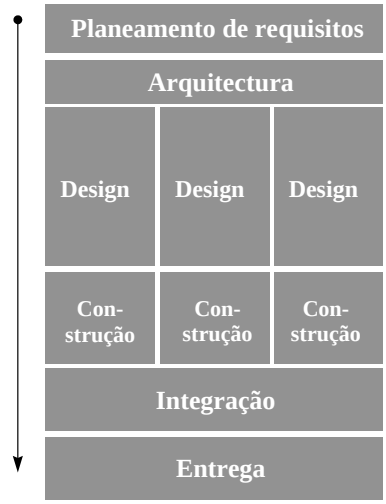


Figura 6.2. As fases do RAD

Durante a *fase de planeamento de requisitos*, os resultados pretendidos do projecto são definidos. Guias gerais para a funcionalidade do sistema são definidas, tal como os requisitos a serem cumpridos pelos produtos a serem entregues. Com base nos resultados a serem atingidos, subsequentemente, um plano de desenvolvimento é arquitectado.

Durante a *fase de desenho*, as funcionalidades do sistema são documentadas. As especificações são desenhadas interactivamente nas *workshops de desenho da aplicação* (*joint applications design workshops* – JAD). Os utilizadores fornecem o *input*, que é representado pelos desenhadores — na forma de especificações numa ferramenta CASE. Os protótipos são criados com a ajuda de geradores de programas. Os utilizadores podem então testar as especificações directamente sobre os protótipos. Nos métodos de desenvolvimento tradicionais, a fase de desenho é claramente distinta da fase de construção. No RAD, não é o caso: o *software*, dentro de certos parâmetros, pode ser gerado a partir das especificações descritas com a ajuda da ferramenta CASE.

Durante a *fase da construção*, o *software* gerado é aperfeiçoado e os elementos que não puderam ser produzidos automaticamente são criados manualmente. A validação do desenho pelos utilizadores continua durante esta fase. Durante a *fase de entrega*, são realizados testes de aceitação e o sistema é preparado para entrar em produção. Isto envolve actividades tais como a instalação, conversões necessárias e formação dos utilizadores. Para aplicações mais extensas, um número limitado de desenhos e caminhos de construção em paralelo podem ser tomados, tendo em mente a gestão do projecto.

A fim de integrar os componentes individuais do sistema entre si, uma arquitectura técnica para o seu relacionamento é desenhada separadamente durante a *fase de arquitectura*, antes do início da fase de desenho.

Quando a construção está completa, a operação de cada componente é testada separadamente durante a *fase de integração*. Este é um teste inicial — dirigido principalmente à compatibilidade técnica entre os diferentes componentes — efectuado antes do sistema ser entregue ao utilizador para um teste de aceitação e execução.

6.2 O Método "IPSD"

IPSD significa *Interactive*, *Process-oriented System Development*. O desenho de processos de negócio eficientes e o desenvolvimento de sistemas de informação para suportá-los são combinados numa aproximação interactiva nos quais sistemas de *workflow* podem ser desenvolvidos interactivamente num contexto BPR. Além disso, este modelo é também aplicável nas situações onde nenhum sistema de *workflow* esteja a ser desenvolvido, e assim nenhum software de gestão de *workflow* esteja a ser utilizado. Entretanto, nesta discussão, assumimos, uma situação em que *workflows* existem tal como foi descrito neste livro.

Se projectarmos as fases do RAD no ciclo de vida do BPR, então o ciclo de vida IPSD de um sistema do *workflow* é gerado. Isto é ilustrado na figura 6.3. Note-se que as fases da figura 6.2 (RAD) estão sobrepostas aquelas ilustradas na figura 6.1 (BPR).

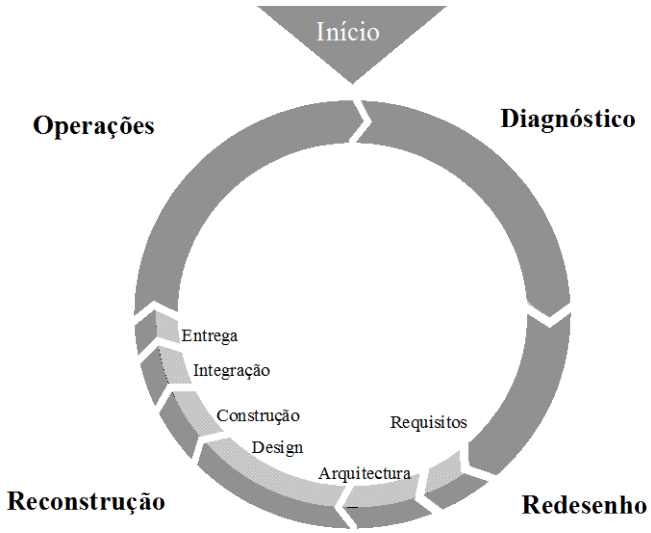


Figura 6.3. O ciclo de vida

No resto deste capítulo, refinaremos este ciclo de vida mais detalhadamente. Finalmente, identificaremos as seguintes fases:

1. preparação;
2. diagnóstico;
3. redesenho do processo;

4. requisitos;
5. arquitectura;
6. desenho de componentes;
7. construção;
8. integração;
9. entrega;
10. execução; e
11. monitorização e melhorias.

Um projecto executado de acordo com o método de IPSD passará por estas onze fases.

Nas seguintes secções examinaremos mais detalhadamente as actividades que são realizadas durante as várias fases. Sendo assim, assumimos o redesenho total de um processo, o desenvolvimento e execução de um novo sistema de informação suportado pelo *software* de gestão de *workflow* conjuntamente com as aplicações de processamento de dados mais "tradicionais". Na secção 6.2.13, concentramos a nossa atenção nas situações em que o *software* de gestão de *workflow* é integrado com sistemas existentes (i.e., sistemas legados).

6.2.1 Princípios básicos

O método IPSD baseia-se no desenvolvimento do melhor processo de negócio possível. Uma boa interacção entre a equipa de desenvolvimento de sistemas de informação e os utilizadores contribui para a sua qualidade e aceitação dentro da organização, além de que também assegura que o seu desenvolvimento prossiga rapidamente e eficientemente. Com base nestas pré-condições, traçamos nove princípios básicos que são essenciais à aplicação bem sucedida do método:

1. O foco está centrado no processo de negócio. Durante todo o ciclo de desenvolvimento, os esforços centram-se em conseguir a melhor estrutura possível para o processo. Entre outros aspectos, significa que um processo de desenho consistente é criado numa fase inicial — com oportunidades para uma melhoria contínua à medida que o desenvolvimento continua.

2. Por definição, a mudança radical que ocorrerá terá consequências para a organização na sua totalidade — ou pelo menos, para parte dela. O sucesso é garantido somente se a gestão (sénior) suportar o projecto e for capaz de transmitir um compromisso inequívoco a toda a organização.

3. Tanto quanto possível, as decisões são feitas pela equipa de desenvolvimento. Isso possibilita que o progresso do projecto tenha um nível de perturbação o mais baixo possível. Os gestores responsáveis devem portanto, ou fazer parte da equipa ou delegar responsabilidades.

4. A equipa de desenvolvimento e os (representantes dos) utilizadores da organização trabalham em equipa para melhorar os processos e desenvolver os sistemas de informação. Juntos são responsáveis pelo resultado. Todos os

participantes respeitam a experiência dos colegas e a contribuição de cada um é tratada da mesma forma.

5. Ao planear e organizar o trajecto do desenvolvimento, a ênfase é colocada nos objectivos do projecto e não tanto na execução (ou na atribuição) das actividades.

6. As especificações do sistema não são inicialmente definidas (e postas em suspenso), mas sofrem uma evolução durante o desenvolvimento. As especificações são inseridas no sistema de *workflow* e numa ferramenta CASE, e testadas com ajuda de protótipos e de simulações (práticas).

7. Os erros são permissíveis durante o desenvolvimento. Devido à natureza iterativa da abordagem, as funcionalidades do sistema são continuamente testadas. Sempre que é detectado um erro, este poderá ser corrigido durante uma iteração futura.

8. A experiência mostra que nenhum sistema é perfeito à primeira. Em vez de dedicar demasiado tempo a procura de soluções (tecnicamente) perfeitas, é melhor conseguir um resultado tangível que seja considerado "bastante bom" num curto espaço de tempo.

9. No fim de cada fase, o planeamento global é actualizado de acordo com informação mais recente.

6.2.2 Preparação

Para a preparação do projecto, é estabelecida uma equipa de projecto. O âmbito e a composição da equipa pode variar durante o projecto, mas inicialmente é desejável começar com um "núcleo" de pessoas que permanecerão envolvidas no projecto até à sua conclusão. Além de um gestor para o projecto, a equipa consiste em representantes das unidades organizacionais envolvidas no projecto. Estes incluem pessoas "da organização do utilizador" e do departamento dos SI/TI, assim como peritos nas áreas de análise e modelação de processos de negócio. A pessoa escolhida para gestor do projecto deve ser alguém com autoridade suficiente dentro da organização. Esta pessoa pode ser alguém da gestão sénior, embora possa haver alguns argumentos contra tal nomeação (tal como a disponibilidade, a falta de conhecimento e as capacidades de trabalho necessárias). Porque um gestor de topo não é necessariamente um bom gestor de projectos, e os gestores de projectos internos (de SI/TI) podem não ter maturidade suficiente, é comum recrutar um gestor de projectos externo.

Dada a importância do projecto e das suas consequências para toda a organização, o objectivo específico do projecto de reengenharia deve ser enunciado o mais claramente possível — preferencialmente pelo nível mais elevado da gestão. Também deve estar absolutamente claro que a gestão apoia totalmente o gestor do projecto e o seu trabalho.

No início de um projecto, o gestor cria um plano do projecto, no qual descreve a abordagem a ser feita e disponibiliza um esboço do calendário. O

objectivo do projecto deve ser, claramente, indicado no plano, e deve haver um relacionamento visível entre a aproximação escolhida e a realização dos objectivos. Por outras palravas, deve ser absolutamente claro como cada actividade contribuirá para alcançar o objectivo do projecto. Nesta fase, o calendário do projecto é ainda aproximado; somente será fixado definitivamente durante a fase do diagnóstico. O plano do projecto será comunicado a todos aqueles que na organização estão envolvidos no projecto. Será dado a conhecer:

Actividades
- Definição da equipa (o núcleo) do projecto;
- Esboço do plano do projecto;
- Obtenção da aprovação para o projecto; e
- Comunicação da missão, abordagem e calendarização.

Entregas
- Plano geral do projecto.

6.2.3 Diagnóstico

Um projecto deverá começar com uma análise da situação existente. Compreender a estratégia existente na organização é uma etapa inicial importante. O *diagnóstico* tem três grupos de actividades: análise, contextualização e visão. Estas estão interligadas e, posto isso, consideramo-las como uma só fase. A *análise* preocupa-se com a situação e a compreensão das razões existentes para a mudança. A *contextualização* é a identificação clara das partes da organização, dos processos, e dos sistemas que devem, ou não, ser consideradas para o projecto. Adicionalmente, deve ser determinado uma agenda para o projecto e um esboço do orçamento. A visão é focalizada nas possíveis vias para a melhoria.

A *análise* começa por procurar as razões para a mudança. Mudança significa transformação ou reengenharia dos processos de negócio, da organização e dos sistemas de suporte de informação. Frequentemente, existem congestionamentos no desempenho dos processos ou nos sistemas de suporte de informação. Estes congestionamentos podem ser de natureza quantitativa, o que significa que os processos têm pouca capacidade para entregar produtos ou serviços para satisfazer os pedidos dos clientes. É também possível que os congestionamentos sejam de uma natureza qualitativa, o que significa que os produtos ou os serviços que são gerados pelos processos não satisfazem as necessidades do cliente. É possível que ambas as causas possam ocorrer simultaneamente. É também possível que o processo de produção seja demasiado dispendioso. Contudo, existe uma possibilidade de não haver congestionamentos, mas que estes sejam esperados num futuro próximo caso nada seja feito. Todas estas razões são sintomas de alguma "doença." Ao analisar um processo, tem de ser prestada atenção, em particular, aos seguintes aspectos:

- actividades sequenciais e burocráticas desnecessárias;
- formação de uma "ilha de computorização";
- necessidade excessiva de formulários e aprovações;

- uso de papel e estipulações redundantes; e
- politicas orientadoras e regras (formais ou informais) que não são observadas ou que pareçam não funcionar.

Se a organização estiver em boa forma, pode ainda haver uma necessidade de mudança se houver algumas boas oportunidades de estender o negócio ou de melhorar a qualidade ou a eficiência introduzindo novas tecnologias.

Uma compreensão clara das razões do projecto, assim como a estratégia existente, são *factores críticos do sucesso* (*Critical Success Factors* – CSF) para a organização e essenciais para um projecto de reengenharia. Que factores determinam o sucesso ou a falha de um projecto? Uma compreensão clara do valor dos vários processos — por outras palavras, a importância da sua contribuição para o desempenho da organização — é importante ao escolher quais devem ser redesenhados. Isto requer o conhecimento da organização, do mercado e da competição. Apesar de tudo, qual é a finalidade de dinamizar o procedimento de administração para processar pedidos e facturas dentro de uma empresa comercial se essa empresa estiver perdendo pedidos como resultado da gestão ineficiente do inventário e de uma pobre estrutura de distribuição?

A análise das razões para a mudança resultará na formulação dos *objectivos* a serem alcançados. Primeiramente, estes serão feitos em termos qualitativos, tais como "os clientes deverão ser servidos melhor" ou "os custos de produção deverão ser minimizados."

De forma a traduzirem os objectivos em alvos concretos, a etapa seguinte é a formulação e a definição dos *indicadores chaves de desempenho* (*Key Perfomance Indicators* – KPI). Estes indicadores devem ser mensuráveis e devem expressar todos os aspectos relevantes do desempenho dos processos e dos sistemas de informação. Por exemplo, o objectivo "os clientes devem ser servidos melhor" poderia ser expresso por dois indicadores de desempenho: o tempo necessário para completar a ordem e a qualidade do produto ou do serviço avaliado pelo cliente. A relação entre os CSFs e os KPIs, é que os KPIs são quantificáveis e expressam os CSFs. Pode haver mais do que um KPI para expressar um CSF e pode haver KPIs que só podem ser relacionados indirectamente a um CSF.

A etapa final da fase de análise é a *medida nula*: a determinação dos KPIs na situação existente. Isto é extremamente importante porque é a única maneira de ver, mais tarde, se o projecto causou melhorias reais. A medida nula será também usada na fase de redesenho, onde os processos novos serão modelados e analisados para ver se as melhorias pretendidas foram atingidas. Esta medida pode ser trabalhosa por não existirem dados na administração ou por não ser fácil obtê-los a partir dos sistemas de informação existentes. É sempre possível usar técnicas de amostragem para obter pelo menos algumas estimativas dos KPIs, por exemplo, seguindo uma amostra de ordens do cliente através dos processos e dos sistemas. De facto, esta amostra pode ser usada mais tarde na forma de casos de utilização e sistemas de tal forma que possam ser re-usados. Os casos de utilização, referidos também como casos de negócio, devem incluir os mais importantes tipos de casos, nomeadamente as excepções e os erros que ocorrem na prática.

Durante a análise torna-se claro quais as partes da organização, os processos e os sistemas de informação que têm de ser transformados de forma a atingir os objectivos. Assim, o *contexto* do projecto decorre em paralelo com a análise.

Frequentemente, existem muito boas razões para limitar o contexto do projecto, embora isto possa implicar que partes relevantes sejam suprimidas. Isto significa que não pudemos encontrar a melhor solução, mas isto pode contrabalançar o risco do projecto se tornar difícil de gerir ou que a continuidade das operações existentes fique em risco. Finalmente, os limites de tempo e de custo são frequentemente estipulados numa fase e por isso requerem restrições de contexto do projecto.

Frequentemente, o processo de análise tem o efeito paralelo de gerar ideias para melhores soluções. A *visão* da situação futura é criada nesta etapa e começa com a opinião de um perito sobre a perspectiva da situação futura. Uma vez que os processos que necessitam de reengenharia já foram identificados, a pergunta a ser respondida agora é como pode ser conseguido o melhor resultado, aplicando tecnologias de informação e de comunicação. Tecnologias modernas, tal como sistema de tratamento de imagens, automatização do trabalho de grupo, gestão de *workflow*, sistemas periciais e de suporte à decisão oferecem oportunidades para estruturar os processos dentro de uma organização de uma forma inteiramente diferente. Frequentemente, é também útil olhar para além dos limites da própria organização. O uso da infra-estrutura da Internet com tecnologias avançadas, tais como a tecnologia Web, troca de dados electrónicos usando o XML, o *e-mail* e os *smartcards*, podem resultar em melhorias notórias. Uma pesquisa nas oportunidades que estas tecnologias oferecem para a reengenharia dos processos requer um conhecimento destas e uma análise aprofundada da sua aplicabilidade. É necessário ter em consideração elementos, tais como a extensão, em que tais tecnologias podem ser incorporadas na infra-estrutura existente.

O desenvolvimento de uma visão de reengenharia de processos de negócio requer uma equipa multidisciplinar que inclua representantes da gestão da organização e peritos em SI/TI. Além disso, está claro que um elevado grau de compromisso da parte dos gestores séniores é uma pré-condição importante. A fim de conseguir a mudança radical pretendida, as ideias "arrojadas" e controversas devem ser uma hipótese a não ignorar.

A medida nula é feita pela equipa do projecto e requer frequentemente um trabalho de pesquisa. A maioria das outras actividades são feitas durante sessões conjuntas de trabalho com os representantes das unidades organizacionais relevantes e se possível com a gestão.

Actividades

Análise

- Análise das razões para a mudança, da estratégia e dos factores críticos de sucesso;
- objectivos a atingir após a transformação, formulados de forma qualitativa;
- definição dos indicadores chave de desempenho capazes de quantificarem os objectivos e medir as melhorias pretendidas; e
- medida nula: determinação dos indicadores de desempenho na situação existente.

Contextualização

- Identificação das partes da organização, dos processos e dos sistemas que devem permanecer inalterados e dos que se inserem no âmbito do projecto; e
- determinação de condições de limites temporais e financeiros.

Visão

- Opinião do perito sobre a nova organização, sobre os processos e sobre os sistemas;
- especificação das metas a serem realizadas no projecto, isto é, quantificação dos objectivos em termos dos indicadores chave de desempenho; e
- geração das ideias e dos guias para o redesenho.

Entregas

- Documento descritivo das razões para a mudança, os objectivos e o KPIs;
- conjunto de casos de utilização;
- a medida nula;
- lista dos processos, das partes da organização e dos sistemas de informação a serem alvo de reengenharia;
- condições de limite no tempo e no custo;
- opinião do perito sobre a nova situação e sobre ideias para melhorias; e
- especificação dos alvos em termos de KPIs.

6.2.4 Processo de redesenho

A fase do redesenho começa com a modelação dos processos existentes. Por duas razões: é uma maneira de compreender melhor os processos existentes oferece a oportunidade de calibrar o modelo da situação existente com a medida nula. Desta forma, podemos estimar os parâmetros dos processos que não serão afectados pelo redesenho. Estes serão usados nos modelos dos processos redesenhados. É também uma verificação: se os congestionamentos e os KPIs computados pelo modelo diferem demasiado dos valores da medida nula. Isso significa que algo está errado: ou a medida nula é errada ou o modelo é errado, o que significa que não compreendemos correctamente a situação existente. Nós defendemos o uso de redes de Petri na modelação. Ferramentas de simulação podem ajudar-nos na computação dos KPIs, embora às vezes métodos analíticos estejam disponíveis.

A partir do momento em que os alvos do projecto de reengenharia tenham sido formulados e a situação existente tenha sido avaliada, a etapa seguinte do projecto pode ser iniciada: o novo processo pode ser desenhado. Neste momento, o projecto cresce. A equipa do projecto é expandida para incluir os utilizadores finais com o conhecimento detalhado dos processos de trabalho existentes e das exigências associadas. A participação intensiva dos utilizadores irá preparar a aceitação das mudanças futuras e permite uma identificação antecipada dos riscos. Além disso, perícia no campo da configuração do *software* de gestão de *workflow* acrescenta valor à equipa do projecto.

A fase de redesenho continua com uma série de *workshops* conjuntos para estabelecer uma base para o redesenho. Representantes das unidades organizacionais envolvidas no projecto participam nestas sessões, junto com a gestão da organização. Geralmente, duas ou três destas *workshops* são suficientes para tratar de todos os tópicos relevantes. Usando os princípios de melhoria mencionados no capítulo 3, vários cenários alternativos para a organização dos processos de negócio são projectados e avaliados. Estes cenários não são (pequenas) variações de um único modelo de processo, mas são variações que diferem fundamentalmente uma da outra na sua aproximação. Por exemplo, o controlo centralizado versus o controlo descentralizado, formas abrangentes de *outsourcing*, uso de EDI, aplicações para a Internet, e assim por adiante. Nesta fase, a descrição dos modelos de processos alternativos estará num nível abstracto.

Para fazer uma avaliação das alternativas tão eficientemente quanto possível, alguns tipos de visualização ou de prototipagem são aconselháveis. Para tal, utilizamos ferramentas específicas para modelar processos de negócio. As ferramentas, baseadas em modelação de redes de Petri são naturalmente favoritas, mas existem outras que poderiam também ser usadas. Muitos sistemas de gestão de *workflow* incluem uma ferramenta de modelação a qual suporta uma forma simples de animação. Dado o grau de abstracção dos modelos de processos, ferramentas que disponibilizam alguma forma de animação são mais apropriadas. Nesta fase, um conjunto de casos característicos são projectados. Devem representar os tipos mais importantes de casos, incluindo erros e excepções. Estes casos são chamados *casos de utilização* e serão usados também nas fases seguintes.

Com base na discussão e nos argumentos propostos durante o(s) *workshop(s)*, uma das alternativas é seleccionada. Esta escolha é então modelada com tanto detalhe quanto possível durante a fase seguinte. Tal desenvolvimento é feito usando o princípio da iteração. Uma proposta inicial é projectada por um perito no campo da modelação de processos, usando preferencialmente uma ferramenta que suporte modelação recorrendo a redes de Petri. Este modelo é melhorado iterativamente e refinado durante uma série de *workshops* onde os casos de utilização são usados para efectuar testes manualmente. Existem ferramentas para suportar a verificação da correcção dos processos. O KPIs dos novos processos que expressam desempenhos logísticos, tais como o *throughput*, os tempos de espera e a utilização dos recursos, podem ser calculados por meio de simulação. Estes são comparados com os objetivos. É também possível determinar o quão sensível é o processo desenhado a problemas internos da empresa (por exemplo, a doença de um funcionário).

A simulação mostra apenas os KPIs logísticos dos novos processos, mas não os KPIs funcionais. Os KPIs funcionais podem ser determinados por meio do ciclo de vida das experiências ou *jogos* com a ajuda de um sistema de gestão de *workflow*. Neste caso, o modelo do processo é executado no sistema de gestão de *workflow* e os participantes no *jogo* desempenham uma situação prática com a aplicação do novo processo. Tal aproximação requer muita preparação, e, devido à sua estrutura, é frequentemente limitada na extensão em que consegue simular todas as possíveis excepções e efeitos do volume de processamento. É, todavia, particularmente um bom instrumento para envolver os utilizadores no desenvolvimento e para encorajar o suporte a futuras mudanças. Por estas razões, pode ser um complemento muito eficaz ao uso da simulação.

O resultado é um novo modelo de processos que serve de base para o desenvolvimento e execução. Como o modelo é melhorado, todos os requisitos e pré-condições relativas aos sistemas de processamento de dados são gerados. Tanto quanto possível, estes são registados e serão usados numa fase mais avançada, quando os sistemas que têm de suportar o processo são projectados e construídos.

O redesenho dos processos terá geralmente consequências extensas para a estrutura da organização. Os limites tradicionais entre os departamentos e as unidades de negócio podem deslocar-se ou mesmo desaparecer. As responsabilidades mudam e a tomada de decisão pode transitar para outro local.

Durante a fase de redesenho, deve ser dada atenção às consequências para a estrutura organizacional e para a gestão dos recursos humanos (*Human Resource Management* – HRM). Os pontos a serem observados são:

- redefinição das tarefas e das funções;
- equipas auto-geridas e as capacidade de gestão associadas;
- sistemas baseados em estimativas;
- estruturas salariais; e
- formação e treino

Estes aspectos são registados num modelo organizacional e usando uma descrição das medidas que seriam necessárias para concretizar este modelo.

Actividades

- Modelação e calibração da situação existente;
- desenvolvimento das alternativas para o novo processo de negócio;
- análise da alternativa seleccionada: determinação das propriedades de correcção e dos KPIs logísticos (por simulação);
- análise dos KPIs funcionais por meio de *workshops* de teste usando sistemas de gestão de *workflow* (opcional); e
- descrição das consequências para a organização.

Entregas

- Modelo calibrado dos processos existentes;
- conjunto de casos de utilização;
- modelos para os novos processos escolhidos;
- resultados dos testes de simulação e dos jogos;
- requisitos para aplicações de processamento de dados; e
- modelo organizacional.

6.2.5 Requisitos

O núcleo do novo sistema de *workflow* foi estabelecido. O sistema de processamento de dados, que tem de suportar o processo, pode ser desenhado e construído. Antes de o fazer temos de estabelecer cuidadosamente quais as funcionalidades que o sistema de processamento de dados tem que albergar de modo a planear e orçamentar as fases seguintes. Isto é novamente conseguido

através de uma série de dois ou três *workshops*. Estes cobrem os seguintes tópicos:

- Modelo de dados dos sistemas. Distinguimos dados associados a casos e dados não associados a casos. Os dados associados a casos são modelados mais adequadamente se forem representados como um dossier que é preenchido durante o processo. Os dados não associados a casos podem ser divididos em dados de suporte e informação de gestão. Dados de suporte, são dados usados no processamento de casos, como moradas, taxas e instruções; a informação de gestão está relacionada com a qualidade e eficiência do tratamento dos casos.
- Interacção entre etapas do processo e aplicações de processamento de dados. O ponto de partida é o modelo do processo: cada tarefa requer alguns dados e produz alguns dados para o dossier. A relação com o processo desenvolvido durante a fase anterior é estabelecida através de uma matriz construída com as etapas do processo e com as funcionalidades do sistema que usaram.
- Funções de processamento de dados suplementares para assuntos como a gestão e a troca de dados.
- Requisitos do sistema a nível da velocidade, capacidades de processamento, flexibilidade, entre outros.
- O desenvolvimento, as estratégias de execução e o agendamento. É verificado se todas as funcionalidades podem ser conseguidas e introduzidas de uma só vez, ou se é preciso adoptar uma estratégia de desenvolvimento incremental.
- Riscos e estratégia de gestão de risco.

Os resultados da fase de redesenho do processo, em particular, aqueles que resultam das *workshops* fornecem uma boa base para desenvolvimentos adicionais. É certo que nem todos os detalhes são conhecidos, mas a visão actual disponível do processo e sistemas a serem desenvolvidos, e dos requisitos que estes devem respeitar, é clara. Dado os assuntos abordados nesta fase, a equipa de projecto é expandida a fim de incluir um ou mais programadores experientes. Estes vão ser envolvidos no estabelecimento do novo sistema durante a fase de desenho e construção.

Com base nos requisitos recolhidos nos *workshops*, o plano geral do projecto, desenhado durante a fase de preparação, pode ser ainda mais aprofundado.

O plano do projecto incorpora todos os tópicos levantados durante os *workshops* de planeamento de requisitos, incluindo uma agenda detalhada das etapas seguintes.

Actividades

- Preparação e planeamento dos *workshops* de requisitos;
- elaboração de medidas de gestão de risco;
- elaboração de uma agenda do projecto e orçamento; e
- desenho de um plano de projecto detalhado.

Entregas

- Rascunho do modelo de dados (entidades e relações);
- rascunho do modelo funcional das aplicações a serem desenvolvidas;
- matriz de funções para cada processo (etapa); e
- plano detalhado do projecto para os passos seguintes.

6.2.6 Arquitectura

Antes de começarmos com o desenvolvimento actual dos sistemas, um grande número de opções técnicas precisam de ser tomadas. Um sistema de *workflow* é um sistema complexo, que pela sua natureza e estrutura é distribuído. É necessária uma boa arquitectura de modo a que todos os *componentes* do sistema trabalhem o melhor possível entre si. Esta "arquitectura" descreve vários componentes do sistema, e descreve a forma como estes comunicam entre si (descrição da *interface*). A este respeito, distinguimos a *arquitectura funcional* da *arquitectura técnica*. A primeira subdivide o sistema em componentes funcionalmente interdependentes. Esta estruturação funcional permite que diferentes equipas possam trabalhar em diferentes componentes em paralelo. A arquitectura técnica subdivide o sistema em componentes de *software* e/ou *hardware*. Esta estrutura é grandemente influenciada pelas tecnologias existentes, configuração do sistema gestor de base de dados, sistema operativo, e outros. A arquitectura funcional e a arquitectura técnica estão, frequentemente, interligadas entre si. Deste modo, uma descrição completa da arquitectura consiste numa descrição, quer funcional, quer técnica e esclarece uma relação entre estas. Os exemplos seguintes são aspectos relevantes na descrição de uma arquitectura:

- Infra-estruturas técnicas (*hardware*, redes, SO, e protocolos de comunicação);
- software de gestão de *workflow*;
- ferramentas de desenvolvimento;
- descrição das interfaces (sistema de gestão de *workflow* versus componentes, componentes versos outros componentes e componentes versus bases de dados); e
- interface gráfica com o utilizador *standard*.

Desta forma, uma espécie de "framework" é definida onde se inserem vários elementos do sistema de *workflow*. A figura 6.4 mostra, graficamente, como a descrição da arquitectura pode prestar assistência no relacionamento entre vários elementos diferentes entre si.

Os melhores resultados são alcançados quando a arquitectura é baseada em *standards* industriais (abertos). Isto aumenta a probabilidade das ferramentas usadas (WFMS, DBMS e ferramentas de desenvolvimento) disponibilizarem o suporte necessário e assegurarem a continuidade para o futuro. Em particular, a descrição das interfaces anteriormente mencionadas inclui a forma como o software de gestão de *workflow* comunica com aplicações de processamento de dados, e como estas últimas comunicam entre si (ver Capítulo 5). Ao traduzir a arquitectura para um conjunto de directrizes de desenvolvimento, passa a ser possível integrar, com algum grau de independência, os componentes

desenvolvidos sem muitos problemas. Estas directrizes cobrem a programação das interfaces entre os vários componentes, tal como o desenho do sistema – em particular, a sua subdivisão em módulos ou objectos separados.

A fase de arquitectura é de natureza, predominantemente, técnica. Por esta razão, durante esta fase é utilizado pessoal com conhecimento técnico profundo sobre aspectos de integração de sistemas de gestão de *workflow* contendo sistemas gestores de bases de dados e *software* de aplicações num ambiente distribuído. Pode ser necessário desenvolver rotinas de software específicas durante esta fase, de modo a possibilitar a integração dos vários componentes da arquitectura.

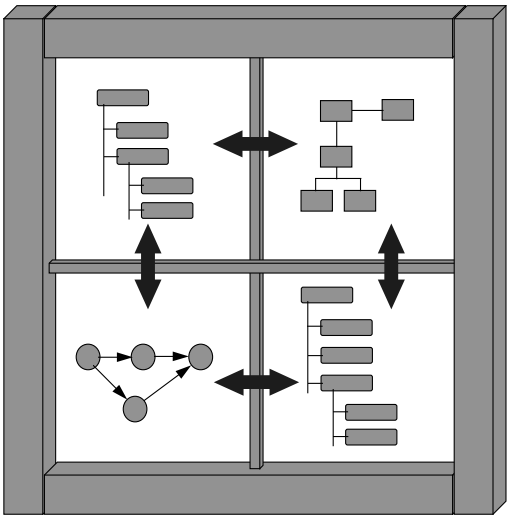


Figura 6.4. Ferramenta de integração

Durante a fase de arquitectura, é recomendado o desenvolvimento de protótipos de vários componentes de modo a que a arquitectura escolhida – em particular as suas interfaces e a sua integração – possa ser testada na prática quanto à sua concretização e, se necessário, melhoradas. Este protótipo pode servir como modelo de referência para desenvolvimentos futuros.

Actividades

- Descrição da arquitectura funcional;
- descrição da arquitectura técnica;
- ilustração da arquitectura técnica e funcional;
- estabelecimento e descrição das directrizes e dos *standards*; e
- desenvolvimento e teste de protótipos.

Entregas

- Descrição da arquitectura;
- protótipo; e
- standards e requisitos dos componentes.

6.2.7 Desenho de componentes

Durante esta fase, a especificação dos componentes de processamento de dados são desenvolvidos iterativamente, usando protótipagem. Os processos especificados durante a fase de redesenho são – se ainda não o foram – implementados num sistema de gestão de *workflow*. O resultado é um protótipo de gestão dos processos. A forma mais rápida de o conseguir é através da existência de uma ligação automática entre a ferramenta de modelação usada durante a fase de redesenho e o sistema de gestão de *workflow*.

Os componentes de processamento de dados são, na sua maioria, criados usando ferramentas CASE e geradores, que posteriormente podem ser rápidamente e facilmente ajustados. Com base nos modelos definidos durante a fase de requisitos, e com o auxílio de geradores de *software*, programadores envolvidos no projecto produzem protótipos dos novos componentes. Integrados com os modelos de processo implementados no sistema de gestão de *workflow*, esses protótipos são submetidos a uma avaliação pelos utilizadores, ao que denominamos de *workshop* de protótipagem. Os protótipos são refinados numa série de ciclos (normalmente três) até corresponderem às necessidades dos utilizadores. Os componentes de gestão de *workflows* (nos WFMS) e de processamento de dados são testados exaustivamente e, se necessário, são ajustados.

São planeadas várias séries de *workshops*. Cada sessão abrange um leque de funcionalidades limitadas, apenas as necessárias para que seja possível a revisão e avaliação pelos utilizadores. O tempo necessário à preparação e ao ajusto dos protótipos também tem de ser tido em conta. Durante os *workshops* iniciais, a principal ênfase vai para o modelo de dados e os *standards* gerais da interface com o utilizador. O número de *workshops* a planear depende do tamanho global do sistema. Aquando da realização do plano de actividades e dos *workshops*, uma parte inteira do processo tem que ser sempre seleccionada: terá de consistir em várias etapas do processo que formem um todo lógico. Isto é necessário de modo a obter, na prática, uma representação realista do *workflow*. Os *workshops* devem ser preparados meticulosamente, tem que ser prestada atenção a assuntos como a divisão clara de papéis e à estrutura de simulação baseada nos casos de estudo.

Graças a este método de protótipagem, não é só a integridade dos componentes de processamento de dados e especificações da definição de processo que são testados, mas também podemos testar a concretização prática do processo.

No fim da fase de desenho, os utilizadores presentes na equipa dão o seu aval formal sobre as funcionalidades. Isto abrange todas as especificações já implementadas no sistema de gestão de *workflow* e as ferramentas CASE (representadas por protótipos), assim como uma lista de futuros trabalhos e/ou ligações com outros componentes. Estes últimos são realizados durante a fase seguinte: construção.

Actividades

- Harmonização do modelo de dados e da interface com o utilizador;
- desenho/geração/harmonização das funcionalidades dos componentes de processamento de dados e definições dos *workflows* usando a protótipagem e simulações de casos de utilização; e

- estabelecimento de especificações para ligações particulares com sistemas de escritório e/ou outros componentes.

Entregas
- *Standards* para a interface com o utilizador;
- especificação do *workflow* dentro do sistema de gestão de *workflow*;
- especificação dos componentes de processamento de dados numa ferramenta CASE;
- protótipo(s) final (finais) do sistema e uma lista de componentes a serem completados; e
- descrição de ligações a fazer com sistemas de escritório e outros componentes.

6.2.8 Construção

Uma grande parte do sistema já está criada (de maneira evolutiva) durante a fase de desenho do sistema. Funcionalidades específicas, que podem ser criadas usando geradores ou que requerem programação adicional, são incluídas durante a fase de construção. Verificações complexas, processamento em lotes e troca de dados com outros sistemas (externos) são alguns exemplos.

As restantes partes do sistema são construídas de forma tradicional, baseadas em especificações não ambíguas.

Finalmente, os vários aspectos do sistema são optimizados para uso no ambiente operacional. Estes incluem:

- A integração do sistema de gestão de *workflow* com aplicações de processamento de dados e aplicações de escritório genéricas (processadores de texto, folhas de calculo, *e-mail* e outros);
- extensão e optimização do sistema para uma utilização em grande escala;
- optimização de desempenho;
- funções de informação de gestão (por não serem incorporadas como *standards*)
- funções de gestão técnica; e
- *software* de conversão.

Embora a fase de construção envolva principalmente aspectos técnicos, o utilizador deve continuar envolvido. Especialmente, ao testar e avaliar os resultados, o *feedback* dos utilizadores activos permanece altamente recomendável. Os utilizadores interessados são também profundamente envolvidos na preparação dos testes de aceitação e execução.

Actividades
- Integração e optimização do sistema de gestão de *workflow*;
- configuração do ambiente de testes;
- completar a documentação do sistema;
- testar o sistema; e
- preparação dos testes de integração e de aceitação.

Entregas

- Componentes prontos para o teste de integração;
- documentação do sistema;
- plano de integração e testes de aceitação (incluindo os casos de utilização); e
- *software* de conversão.

6.2.9 Integração

Por definição, um sistema de *workflow* consiste em vários componentes. A gestão dos processos, implementada pelo *software* de gestão de *workflow*, é uma unidade operacional independente com a sua própria dinâmica e seu próprio ambiente de gestão, e em muitos casos, o seu próprio ambiente de *hardware*. Os componentes separados comunicam através de interfaces. O plano arquitectónico destes componentes e as suas inter-relações são obtidos durante a fase de arquitectura. No caso especial de grandes aplicações, o desenvolvimento de programas desenrola-se em sub-projectos mais ou menos independentes. É importante manter um certo nível de autonomia para estes sub-projectos de modo a atingirem a sua conclusão. Consequentemente, o grau de bom funcionamento entre componentes depende da qualidade e detalhe da arquitectura definida. É durante o teste de integração que verificamos se os componentes separados são mutuamente compatíveis.

Este teste centra-se principalmente nas operações das funções em combinações (técnicas) umas com as outras, e em particular nas interacções entre os vários componentes. Aqui os casos de utilização, desenhados na fase de redesenho, são reutilizados. Este conjunto é estendido e forma a base dos *scripts* de teste.

A actividade mais abrangente consiste em determinar se as funções criadas funcionam devidamente e se produzem os resultados esperados em todas as circunstâncias. É dada mais atenção aos pontos de contacto entre os diferentes componentes: as interfaces. Além disso, assuntos como segurança e autorização, desempenho (picos de carga, carga de longo termo), e recuperação (*recovery*) são testados. Naturalmente, qualquer falha ou erro que possa ocorrer, durante os testes, deve ser corrigido o mais rapidamente possível.

Para uma boa avaliação do comportamento dos componentes, é vital que o teste de integração seja levado a cabo numa infra-estrutura de *hardware* e *software* idêntica ao ambiente de produção final, ou o mais próximo possível. Esta condição vai evitar surpresas indesejáveis e entraves inesperados quando o sistema de produção for estabelecido.

Na realidade, o teste de integração é o primeiro passo para a aceitação do sistema, com principal atenção para a compatibilidade técnica e robustez. Durante esta fase, técnicos de informação e utilizadores trabalham em conjunto a fim de entregar, um sistema operacional que possa ser sujeito a um teste de aceitação, (funcional) feito somente por utilizadores.

Actividades

- Conversão de teste;
- realização do teste de integração;

- rectificação de falhas; e
- elaboração do relatório dos testes.

Entregas
- Ambiente e *software* preparado para o teste de aceitação;
- *scripts* de teste (para testes futuros); e
- relatório de testes.

6.2.10 Entrega

O sistema de *workflow* está neste momento tão avançado que pode ser formalmente entregue aos utilizadores. O objectivo do teste de aceitação é verificar se o sistema opera em concordância com as especificações e se respeita todos os requisitos, de modo a suportar da melhor forma possível, os processos de negócio do dia-a-dia. Isto inclui a condição que a organização deve, da melhor forma possível, ser capaz de efectuar o teste de aceitação ao sistema de *workflow* de forma independente.

Por esta razão, os programadores envolvidos no projecto permanecem num segundo plano, dando apoio unicamente quando for absolutamente necessário – por exemplo, no caso de um funcionamento incorrecto de um ou mais componentes.

Um teste de aceitação está direccionado para os seguintes aspectos:

- Funcionalidade (interface com o utilizador, entradas, processamento interno, saídas);
- utilização diária do sistema do ponto de vista dos casos de utilização identificados durante a fase de redesenho;
- gestão (diária); e
- documentação do sistema entregue.

A grande maioria das funcionalidades e a generalidade das funções de gestão do sistema de *workflow* já foram testadas pelos utilizadores durante as fases anteriores. Tais testes são uma parte integral do processo de desenvolvimento, envolvendo sempre os utilizadores durante a criação e avaliação dos protótipos. A essência do sistema já foi cuidadosamente verificada durante o teste de integração. Posteriormente, o teste de aceitação, deve concentrar-se principalmente na utilização do dia a dia e na gestão do sistema de *workflow*, bem como na documentação técnica e documentação para o utilizador.

Nesta situação, a melhor abordagem é sistemática, onde o processo é acompanhado passo a passo usando casos de utilização predefinidos. Para cada etapa do processo, é escrito um *script* de teste, descrevendo as operações que o utilizador deverá desempenhar e os resultados esperados desse teste. Desta forma, é simulada da melhor forma possível a utilização diária e as operações dos processos pode ser avaliada.

Para além da aceitação funcional descrita anteriormente, deverá ser feito um teste de aceitação técnico por futuros gestores do sistema. Neste teste, são feitas verificações para determinar se o *software* produzido está de acordo com

standards e normas de qualidade gerais que foram estabelecidas para o projecto.

Actividades
- Realização do teste de aceitação utilizando cenários;
- correcção de falhas; e
- produção de um relatório do teste de aceitação.

Entregas
- Ambiente e *software* prontos para serem utilizados e geridos;
- aceitação formal pela organização cliente;
- aceitação formal pela organização gestora; e
- relatório do teste de aceitação.

6.2.11 Execução

A reestruturação de todo o processo de negócio e a execução de novas tecnologias têm consequências na forma como as pessoas trabalham (juntas). As relações hierárquicas tradicionais alteram-se ou deixam de existir, e as responsabilidades mudam. Isto coloca exigências, não só aos processos e sistemas de informação, mas também às pessoas que com eles trabalham. As exigências, com respeito ao conhecimento e às competências, mudam quer no campo social quer no campo técnico.

A execução de um sistema de *workflow* numa organização é tão importante como o seu desenho e construção. Quais as expectativas dos funcionários? Será que estão bem preparados para as suas novas tarefas? Possuem o conhecimento e competências necessárias? As ferramentas disponíveis são suficientes? Foram obtidos todos os acordos necessários?

A execução requer uma preparação minuciosa e o interesse explícito no projecto. Preferencialmente, uma equipa especial deve ser inserida na organização do projecto, para lidar com a preparação e subsequente supervisão. Isto implica que uma parte considerável do orçamento do projecto seja dedicada às actividades de execução.

As actividades da equipa de execução fazem de alguma forma "sombra" às outras fases da abordagem IPSD. Ainda na fase de redesenho, a equipa deve preocupar-se com a análise das implicações do projecto para a organização e questões relacionadas com a gestão de recursos humanos. À medida que o projecto progride, a atenção concentra-se no que é necessário preparar para uma execução bem sucedida. Isto inclui fornecer informação sobre o projecto e os seus resultados (em particular, as mudanças que a organização deve esperar), bem como a preparação de material de formação, elaboração de cursos e monitorização contínua da organização, depois do sistema ter sido implementado.

A equipa de execução deve ser, preferencialmente, composta por funcionários que conheçam bem a organização e que tenham bons contactos. Para levar a cabo as actividades descritas anteriormente, a equipa deve ser composta por pessoas capazes de desempenhar as seguintes funções:

- Perito em comunicação;
- perito na elaboração de documentos técnicos;
- perito organizacional;
- perito de infra-estruturas;
- instrutor; e
- supervisor de processo.

Idealmente, tal equipa compreenderá representantes de utlizadores da organização, dos SI/IT, funcionários executivo e – possivelmente – peritos externos.

Actividades
- Comunicações sobre o progresso do projecto;
- comunicações sobre futuras alterações;
- descrição da estrutura organizacional;
- preparação da descrição de casos;
- preparação de manuais;
- preparação de material para instrutores;
- fornecimento de instrução;
- planeamento e execução da infra-estrutura técnica;
- preparação e supervisão de conversões; e
- supervisão do processo de mudança.

Entregas
- Plano de execução;
- plano de comunicação;
- plano de conversão;
- modelo organizacional;
- descrição de casos;
- manuais;
- informação e material de instrução; e
- infra-estruturas.

6.2.12 Monitorização e melhorias

Uma vez implementado o sistema de *workflow* com sucesso, resta verificar se os melhoramentos pretendidos foram alcançados e se podem ser mantidos. Isto requer a permanente monitorização dos processos usando os critérios de desempenho predefinidos. Estes são chamados de *Key Performance Indicators* (KPIs) estabelecidos durante a fase de diagnóstico. O sistema de gestão de *workflow* pode ser útil na medição e na avaliação destes últimos. Porque o sistema guarda uma grande quantidade de informação sobre os processos e os casos individuais, na prática, é fácil obter uma visão geral do comportamento e desempenho dos processos. Estes indicadores são principalmente indicadores "hard", tais como a utilização do sistema, os tempos de processamento, carga de trabalho, fontes de trabalho e produtividade. Adicionalmente, podem ser

efectuadas outras pesquisas tais como o nível de serviço, satisfação dos clientes e a qualidade. Isto pode ser considerado um seguimento da fase de diagnóstico, já em execução, com o objectivo de identificar potenciais melhorias. Pode sugerir ajustes aos processos e aos sistemas a ele ligados – as mudanças não são tão radicais como no BPR, são geralmente melhorias mínimas nos processos.

Chamamos a esta abordagem *Continous Process Improvement* (CPI). Porque as mudanças não são demasiadas, a frequência com que podem ser implementadas é muito maior. A figura 6.5 ilustra o posicionamento relativo do CPI e do BPR.

A utilização de *software* de gestão de *workflow* tem claras vantagens a este respeito. Porque a definição dos processos é estabelecida em termos de parâmetros, o ajustamento do processo requer um esforço relativamente reduzido, o que torna a tomada de decisão mais simples. Consequentemente, surge um processo de medição, redesenho e execução virtualmente contínuo. A abordagem IPSD também pode ser usada como linha geral de orientação no CPI, partindo do princípio que as actividades têm um domínio restrito e que são executadas de forma sequencial e repetitiva. Por vezes, podemos "saltar" actividades, nesse caso não há necessidade de uma separação entre as fases. Mas as listas de actividades e produtos utilizados no método IPSD constituem uma boa fonte de controlo (*check-list*) para o planeamento e implementação de tais projectos.

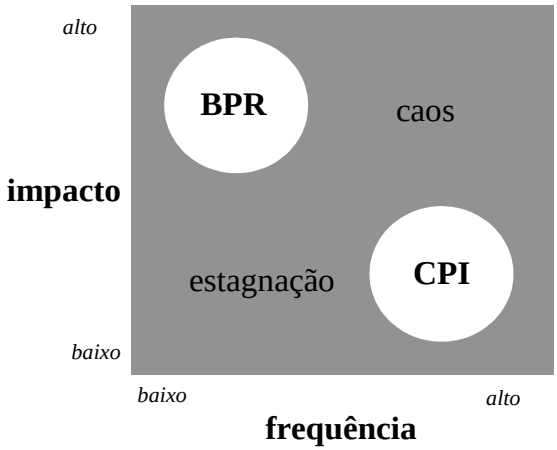


Figura 6.5. BPR versus CPI

6.2.13 Integração de WFMS com sistemas legado

A descrição do método IPSD, feita anteriormente, assume que o novo sistema de informação é desenvolvido lado a lado com os novos processos. Embora, em muitos casos, sistemas existentes (também) têm de estar integrados no *software* de gestão de *workflow* para criar o sistema de *workflow*. Na realidade, isto proporciona uma boa oportunidade para dar uma nova vida a sistemas desactualizados e difíceis de manter.

Em termos gerais, o método IPSD adequa-se bem a tais situações. Contudo, surgem alguns problemas específicos que precisam de ser tratados.

Durante a integração de um sistema legado existente, é necessário manter alguns componentes em vez de criar novos. Em vez de gerar protótipos, é necessário realizar actualizações intensivas no *software* já existente (e muitas vezes ultrapassado). O ambiente de desenvolvimento no qual esses programas foram construídos não é totalmente compatível com o tipo de prototipagem usado no método IPSD. Como resultado, as fases de desenho e construção, em particular, devem ser estruturadas de uma forma diferente. Os componentes já existentes podem, na realidade, servir como um protótipo inicial, mas interacções que resultem numa protótipagem rápida e sequencial não é possível. Não obstante, pode ser utilizada uma forma de desenvolvimento evolucionário. Se as adaptações à interface com o utilizador forem limitadas então um *software* desactualizado não é um grande obstáculo.

Se as modificações são de natureza mais abrangente podemos optar por instalar um ambiente de programação mais moderno para a interface.

Reconstruir partes do sistema "do zero" muitas vezes demonstra ser mais barato do que realizar alterações extensas no *software* existente – especialmente quando se incluem no cálculo a manutenção a longo prazo.

Outro aspecto relacionado com a integração de sistemas já existentes é a eliminação de aspectos de *workflow* antigos de aplicações legadas. Muitos programas, mais antigos, contêm funcionalidades que suportam algum tipo de *workflow*. É vantajoso remover tais funcionalidades tanto quanto possível das aplicações legadas e implementá-las no sistema de gestão de *workflow*. Isto reduz a quantidade de esforço necessário para manter o sistema legado e permite-nos beneficiar de imediato das vantagens da flexibilidade proporcionada pelo sistema de gestão de *workflow*. Durante a fase de arquitectura tem de ser cuidadosamente estabelecido que partes do sistema existente podem ser removidas e como é que o sistema de gestão de *workflow* e aplicações legadas comunicam entre si.

Um problema mais sério é a "incompatibilidade" entre as etapas do processo e a arquitectura do sistema das aplicações existentes. A modularidade desses programas não corresponde às etapas do novo processo desenhado o que complica a interacção entre o sistema de gestão de *workflow* e as aplicações de processamento de dados. Etapas separadas são definidas no processo. Embora cada uma destas corresponda a diferentes funções elas são todas implementadas através de um único programa abrangente, por exemplo em COBOL. Em tais casos, é virtualmente impossível invocar funcionalidades das aplicações existentes no sistema de gestão de *workflow*, mesmo quando essas funcionalidades existem. A figura 6.6 ilustra esta situação graficamente.

A solução para este problema tem de ser obtida de forma a que o código existente seja encapsulado *(rewrapped)* – preferencialmente em pequenas unidades que permitam suportar interacções entre as etapas do processo e as funcionalidades do sistema legado. Esta técnica é denominada de *object wrapping*. Ao definir interfaces intuitivas, o desenvolvimento de standards para ambientes distribuídos e arquitectura de objectos, tais como DCE e o CORBA, contribui para a reutilização de *software* existente. Considerar a utilização de produtos baseados nestes *standards* faz parte da fase de arquitectura. Então, o código do sistema existente pode – numa etapa separada, entre as fases de

arquitectura e de desenho ser reestruturado e encapsulado de tal forma que se torna possível uma reutilização flexível desse código já existente.

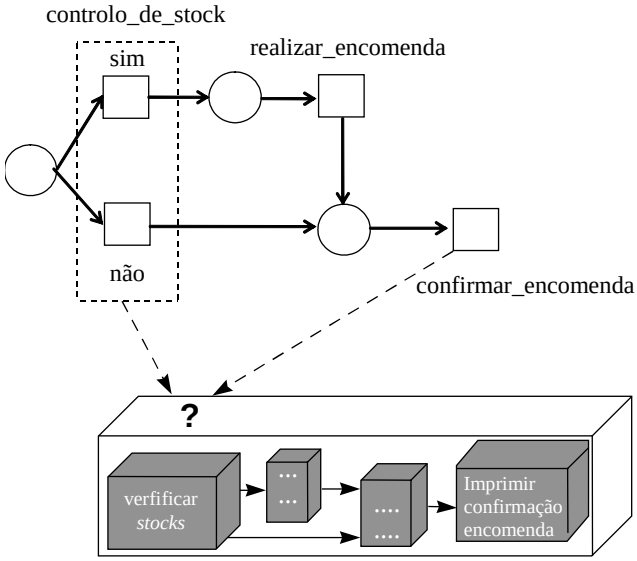


Figura 6.6. Modularidade de aplicações legadas

Enterprise Application Integration (EAI) emergiu como sendo o mais recente gestor de informação. A sigla EAI identifica e interliga *workflows* e funções de aplicações através de sofisticadas filas de mensagens e tecnologias baseadas na Web. As ferramentas EAI identificam, capturam, integram e disponibilizam dados e funcionalidades do sistema aos utilizadores através de uma série de interfaces multi-plataforma e multi-funcionalidades. A tecnologia de filas de mensagens, de vários fornecedores, amadureceu ao ponto de poder suportar a integração de tais funções sem ser necessário um redesenho ou uma modificação significativa de ambientes legados complexos.

EXERCÍCIOS

Exercício 6.1

(a) Dê duas boas razões para o envolvimento de (potenciais) utilizadores nas actividades do ciclo de vida do IPSD.
(b) Apresente os três critérios de selecção de utilizadores para fazerem parte da equipa de redesenho.

Exercício 6.2

Os casos de utilização têm um papel importante no ciclo de vida do IPSD. Indique onde são utilizados e porque são importantes.

Exercício 6.3

Os requisitos e a arquitectura são duas fases separadas no ciclo de vida do ISPD. Estas poderiam ser integradas numa só fase, normalmente também denominada de "arquitectura". Nesse caso, ambos os aspectos funcionais e técnicos são considerados numa só fase. Indique quais são as vantagens e desvantagens de as ter separadas.

(Página deixada propositadamente em branco)

Capítulo 7.

Caso de Estudo: Sagitta 2000

7.1 Contextualização

Os conceitos introduzidos neste livro apenas podem ser correctamente entendidos quando são ilustrados usando um caso de estudo real. O desenvolvimento do sistema de processamento de declarações Sagitta 2000 do Serviço Alfandegário Holandês, uma divisão parte da Autoridade de Impostos Holandesa, é um excelente exemplo para efectuar o estudo de um caso real. O desenho do novo sistema começou no início de 1995 e chegou agora ao início da fase de construção. Um dos princípios fundamentais no desenvolvimento do Sagitta 2000 sempre foi que – através do processo de desenho e construção – a gestão dos processos complexos de negócio administrativos do Serviço Alfandegário, de que o sistema tratará, seja separada das aplicações que os suportam. Por esta razão, uma atenção cuidada foi dada à modelação da estrutura dos processos durante o desenho do Sagitta 2000. Posto isto, foi verificado que os processos de negócio deveriam eventualmente ser incorporados numa camada de gestão separada.

Este capítulo começa com uma breve descrição das tarefas da organização alfandegária e, em particular, o papel do processamento de declarações. Esta descrição esclarece os problemas que o Serviço Alfandegário enfrenta e que eventos mais significativos ocorreram para implicarem uma necessidade de rever os processos de negócio relacionados com declarações. Devemos também examinar a forma com que os processos de negócio são descritos no Sagitta 2000 e as ideias subjacentes a estas descrições. Subsequentemente, discutiremos a descrição de uma parte do processo de negócio do Serviço Alfandegário. No projecto Sagitta 2000, ao longo da fase de desenho também foi efectuada uma pesquisa intensiva sobre como o conceito de gestão podia ser tecnicamente atingido. Isto tornou possível examinar a execução dos processos num sistema de gestão de *workflow*, e os problemas técnicos que surgem da integração de um sistema de gestão de *workflow* com applicações de *software*. Concluímos o capítulo com a revisão de algumas das experiências adquiridas ao longo do projecto e algumas ideias para o futuro.

7.2 Processo de Negócio do Serviço Alfandegário

O Serviço Alfandegário Holandês executa um certo número de tarefas que estão intimamente ligadas ao fluxo de bens que entram e saem da Holanda. Estas incluem a cobrança de taxas Holandesas e Europeias que têm de ser pagas quando se importam bens para a União Europeia. O Serviço Alfandegário também assegura que bens que ponham em perigo a saúde e a segurança da sociedade em geral não entrem no país. Na execução de todas estas tarefas, é vital que o Serviço Alfandegário seja capaz de controlar o fluxo de bens e efectuar verificações selectivas. Isto é feito usando as declarações das várias partes envolvidas no fluxo de bens, declarações que têm de ser submetidas e aceites pelo Serviço Alfandegário. O processo de negócio do Serviço Alfandegário centra-se primariamente no processamento destas declarações.

Porquê redesenhar o processo de negócio?

Desde há muito tempo que o processamento interno das declarações pelo Serviço Alfandegário concentra unicamente num só tipo, dos muitos tipos, de declarações submetidas. Os actuais sistemas de informação do Serviço Alfandegário também estão configurados para lidar principalmente com um tipo particular de declarações. Dois desenvolvimentos significantes estão agora a mudar esta imagem tradicional. Por um lado, o Serviço Alfandegário está a tentar basear o controlo na inspecção do fluxo de bens, assim como no processamento de declarações, e na opinião das partes envolvidas nestes fluxos. Por outro lado, uma nova lei (o Código Alfandegário da Comunidade, ou CAC) resultou na necessidade de adquirir sistemas mais transparentes à forma como as declarações estão relacionadas entre si (o "controlo de bens") e como podem ser realizadas. Estes dois desenvolvimentos alertaram para o redesenho do processo de negócio no âmbito do projecto Sagitta 2000, com o objectivo de criar um procedimento uniforme que pode ser usado para lidar com todo o tipo de declarações.

Porquê separar a gestão e as aplicações?

A manipulação das declarações alfandegárias é um processo que envolve uma enorme quantidade de dados. Controlar e gerir tal quantidade de informação requer uma grande atenção aos detalhes. A separação consciente da gestão dos processos de negócio das aplicações no contexto do Sagitta 2000 possibilita obter:

- Uma oportunidade criada para melhorar o controlo dos processos de negócio (gestão e monitorização). Ao tornar isso explícito é então possível definir a forma como o controlo dos processos deve ser estruturado. Considere, por exemplo: a autorização; a distribuição de trabalho e gestão da carga de trabalho; a separação de funções; e monitorização de progresso. Adicionalmente, torna-se também possível efectuar tanto a gestão como a monitorização dos processos usando sistemas de gestão de *workflow*.
- A garantia que o número formal de passos que têm de ser executados no processo de negócio estão de acordo com a lei. É também recomendável que, por um lado, estes passos possam ser tomados de um modo uniforme por todo o país e, por outro lado, que as várias unidades organizacionais

sejam livres de estruturar o processo dentro dos parâmetros legais, em função das suas necessidades.

- A capacidade de adaptar os processos de negócio às novas necessidades organizacionais e às mudanças na lei de forma mais fácil do que era possível até agora (tudo isto, claro, sem incorrer em elevados custos de manutenção).

Redes de Petri para o desenho de processos de negócio

Como já referimos, o projecto Sagitta 2000 envolveu o redesenho dos processos de negócio para o processamento de declarações alfandegárias. Contudo, no início do projecto, não era ainda claro de que forma a separação da gestão e aplicações iria ser alcançada, nem de que forma o redesenho dos processos de negócio iria ser estruturado. Eventualmente, foi decidido usar redes de Petri para representar os processos de negócio. Isto permitiu modelar um número de características importantes no processamento de declarações de uma forma apropriada:

- Os processos de negócio do Serviço Alfandegário consistem num grande número de tarefas ou etapas individuais (atómicas). Por outras palavras, uma tarefa ou é executada de uma só vez ou não é executada. Algumas tarefas são realizadas por um oficial alfandegário, possivelmente com o suporte do sistema, enquanto que outras tarefas são completamente computorizadas.
- Não existem procedimentos fixos para o processamento de todos os tipos de declarações alfandegárias. Cada declaração tem de ser encaminhada pela rota correcta ao longo do processo de acordo com o seu conteúdo (os seus atributos). Algumas vezes é necessário fazer uma escolha entre opções de processamento, após esta escolha, o processo retoma um percurso normal.
- Porque muitas tarefas são iniciadas por eventos que têm origem no contexto do Serviço Alfandegário, é difícil prever as tarefas que serão executadas. Isto significa que o passo correcto a tomar só pode ser determinado após ter ocorrido um particular evento. Este aspecto apenas pode ser modelado adequadamente se os estados "pendentes" nos quais o processo pode aguardar, enquanto espera por um evento específico, possam ser modelados explicitamente.
- Os passos no processo de negócio podem ser activados por vários tipos de disparos. Portanto, é necessário diferenciá-los durante a modelação.
- O "paralelismo" é possível no processamento de declarações. Por outras palavras, dois ou mais sub-processos podem ser realizados independentemente, com uma subsequente sincronização quando o processo regressa ao caminho comum.

Metodologia do Sagitta 2000

O Sagitta 2000 usa redes de Petri muito similares às descritas anteriormente neste livro. Existem, no entanto, algumas pequenas diferenças nos símbolos usados na metodologia do Sagitta 2000. O número de níveis de aninhamentos está reduzido a dois e não são usadas pré-condições. A tarefa (chamada "tarefa de processo" no Projecto Sagitta 2000) é o coração do sistema, e é simbolizada por um rectângulo. "O princípio da agregação de tempo, local e operação" é aplicado a cada tarefa. Os estados em que um caso pode aguardar entre várias

tarefas é ilustrado no Sagitta 2000 por um triângulo invertido. Contudo, o seu significado não é diferente daquele atribuído às condições (lugares) que vimos anteriormente neste livro. O Sagitta 2000 também diferencia os vários tipos de disparos: uma mensagem recebida (símbolo do "envelope"), um momento fixo no tempo (símbolo do "relógio"), automático (símbolo das "rodas dentadas"), ou activação manual por parte do utilizador. De facto, foram identificados seis tipos de disparos. Um exemplo de uma rede de Petri usada no Sagitta 2000 está ilustrado na figura 7.1.

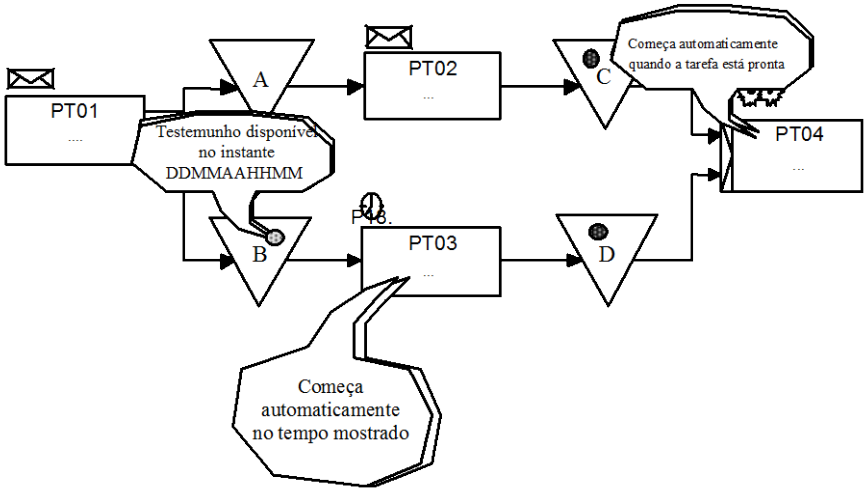


Figura 7.1. Exemplo de uma rede de Petri usada no Sagitta 2000

Relação com aplicações de software

Como referido anteriormente, as tarefas de um processo de negócio podem ser suportadas por aplicações de software. Por outras palavras, uma vez activada uma tarefa, a aplicação, que a executa ou assiste o utilizador a efectuá-la, tem de ser iniciada. Esta aplicação de suporte a tarefas é chamada de tarefa de aplicação. A maioria das tarefas do Sagitta têm uma aplicação associada, mas algumas são totalmente manuais não tendo uma aplicação.

A camada de gestão, introduzida com a adopção de um sistema de gestão de *workflow*, indica à camada de aplicação quais as aplicações que devem trabalhar num caso específico. Depois, a aplicação processa o caso e o seu conteúdo e – uma vez completada – informa à camada de gestão dos (possíveis) ajustes aos valores dos atributos do caso. O valor dos atributos vai ser usado, pela camada de gestão, para decidir quais os estados para os quais o caso deve ser encaminhados e determinar quais as tarefas subsequentes que devem ser iniciadas. Este princípio é ilustrado na figura 7.2.

Uma tarefa, possivelmente associada a uma aplicação, é vista como uma "unidade lógica de trabalho" (ULT) que, ou é executada na totalidade ("commit") ou não é executada ("rollback"). Se uma tarefa é interrompida a meio da sua execução, o estado do caso têm de ser "rolled back" para aquele que existia no momento em que a tarefa começou.

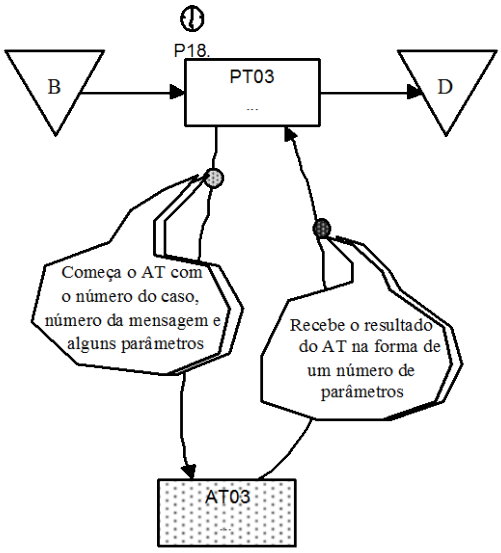


Figura 7.2. Comunicação entre a gestão e uma aplicação

7.3 Métodos de Trabalho

Os processos de negócio do Sagitta 2000 centram-se sempre no processamento de um tipo de caso. Portanto, o projecto começou por determinar os diferentes tipos de casos que podiam ser identificados no contexto do "processamento de declarações". Depois, os processos de negócio foram desenhados para cada um destes tipos de casos. Um processo de negócio é uma sequência de passos (tarefas de um processo) desenhados para processar um caso de um tipo em particular. Cada passo tem de acrescentar valor à sequência e executar as operações necessárias que alteram os atributos de um caso.

Os critérios para desenhar os processos de negócio são sempre aplicados de forma precisa. Por outras palavras, se são estabelecidos critérios, durante a execução de uma tarefa de um caso, que – devido ao seu conteúdo – requer que sejam executadas operações que também são executadas noutro casos que pode ou não ser de um tipo diferente, então estas operações nunca são modeladas como parte integrante de uma tarefa. Tais situações são modeladas pela produção de disparos que resultam do processamento de outros casos que levam à activação de outras tarefas que tratam de casos relacionados. A relação entre dois processos de negócio *nunca* é por isso mostrada criando condições (i.e., lugares) ou tarefas. Se existir interdependência entre diferentes casos, estes são feitos no nível da aplicação. Desta forma, a execução de uma aplicação para um caso em particular pode provocar a produção de disparos para outros casos. Estes disparos não são produzidos pelo sistema de gestão de *workflow*, porque é necessário o conhecimento do conteúdo das declarações para determinar as relações entre casos.

7.3.1 Desenho iterativo

O desenho de um processo de negócio é feito iterativamente. Por outras palavras, não é possível desenhar um processo de negócio num único passo. O desenho inicial, rudimentar é gradualmente refinado através de interacções entre os peritos das alfândegas e os desenhadores. Os peritos das alfândegas, cuja experiência é puramente em técnicas do Serviço Alfandegário, parecem ser altamente capazes de visualizar os seus processos de negócio em termos de redes de Petri. O desenho do processo inicial é rudimentar e é produzido seguindo-se a uma análise dos procedimentos actuais e das leis das alfândegas. Realiza-se também uma sessão de *brainstorming* que estabelece que eventos ocorrem no ciclo de vida de um caso. A nova lei das alfândegas, o Código Alfandegário da Comunidade (CAC), forneceu um excelente ponto de partida. O CAC declara explicitamente que procedimentos estão disponíveis para processar as declarações e quais são os "estados" e as "operações" mais importantes que podem ser identificadas no ciclo de vida de uma declaração.

7.3.2 O que é uma tarefa?

Dentro dos processos de negócio, a tarefa é a mais pequena unidade de trabalho. O critério mais importante para decompor uma tarefa é que tem de *agregar tempo, local e operação*. Porém, durante o desenho dos processos, este princípio nem sempre providencia bases suficientes. De facto, actua como uma espécie de condição básica que, subsequentemente, permite que várias decisões de desenho sejam feitas. O critério não actua imperativamente, no sentido em que uma colecção de operações e funções do sistema têm de ser agrupadas numa única tarefa quando existir pelo menos um procedimento em que a agregação do tempo, local e operação é aplicada a essa colecção. Em tal situação, é legítimo dividir esta tarefa em duas, sendo uma executada imediatamente após a outra.

Uma consideração apropriada também requer que outros critérios sejam levados em conta:

- *Reconhecimento da tarefa*. Para uma organização, uma tarefa, tem de ser reconhecível e envolver um conjunto de operações e funções úteis do sistema. No fim de contas, uma tarefa tem uma função clara e objectiva, e é também a unidade de trabalho para os funcionários. Este último (de forma a separar funções, por exemplo) pode ser um motivo para dividir uma tarefa em subtarefas a serem executadas por diferentes funcionários.
- *Estados interinos sensíveis*. Todos os estados interinos (condições) do processo de negócio devem ter nomes (razoavelmente) sensíveis. Se isto for impossível ou muito difícil, então deve indicar que foi definido um estado que não é reconhecível, ou importante, para os utilizadores.
- *Um "commit work"aceitável para cada uma das tarefas do processo*. A divisão das tarefas e a introdução de um estado interino resulta na criação de um "commit work" separado para cada tarefa. Por um lado, isto provoca flexibilidade para o utilizador; por outro, na situação operacional, deixa de ser possível voltar à primeira tarefa uma vez iniciada a segunda.

7.3.3 Lidar com a complexidade

Os processos do Serviço Alfandegário são demasiado complicados para serem representados e ilustrados numa rede de Petri simples e plana. A descrição de um processo com demasiadas tarefas, condições e caminhos interligados – com um diferente conjunto de exigências associadas a cada caminho – deixa de ser reconhecível e compreensível para os analistas e peritos alfandegários. Além disso, num modelo tão complexo a possibilidade de ocorrerem erros de modelação é muito grande. No Sagitta 2000, foi usada a decomposição hierárquica para ultrapassar a complexidade do processo. Visto que demasiados níveis de decomposição são também difíceis de gerir, o desenho final inclui apenas dois destes níveis. Adicionalmente, foram introduzidas "tarefas de encaminhamento". Estas são tarefas nas quais vários sub-processos convergem, todas as regras de decisão são avaliadas ao mesmo tempo, e o caminho subsequente é determinado.

7.4 Exemplo: Um Processo de Negócio do Serviço Alfandegário

No Sagitta 2000 distinguimos vários processos de negócio, cada um com características totalmente diferentes. Devido à sua forte relação com o fluxo de bens, alguns são críticos em relação ao tempo. Estes incluem, por exemplo, o processamento (*standard*) de declarações. Visto que para cada remessa é necessário realizar uma declaração, existe um grande número de casos. Por outro lado, alguns processos não são críticos em relação ao tempo e envolvem muito menos casos, cada um destes casos pode ser muito extenso. Estes incluem, por exemplo, o processamento das declarações mensais em que a maioria das organizações justifica um mês inteiro de transacções.

Os vários tipos de processo responsáveis pelo processamento de declarações têm em comum uma forte estruturação e complexidade. Dado o facto do Serviço Alfandegário ter constantemente que responder dinamicamente aos eventos gerados em seu redor – que nem sempre podem ser previstos antecipadamente – é vital incluir condições na estrutura do processo. De seguida, descrevemos um exemplo das práticas alfandegárias no que concerne ao processamento de uma declaração *standard*. Primeiro é apresentado o diagrama do processo principal que mostra a estrutura geral do processo. Depois mostramos um sub-processo contendo uma descrição do mais baixo nível da decomposição: a tarefa.

O processo principal

O processo de declaração principal está ilustrado na Figura 7.3. Este é um processo genérico para lidar com todo o tipo de declarações e procedimento de declarações. As declarações são encaminhadas correctamente ao longo do processo usando regras de decisão. A figura 7.3 não mostra a versão mais recente do processo. O Sagitta 2000 é um processo em contínua evolução e o processo de declarações ainda está sujeito a pequenas revisões.

Dentro do processo principal podem ser identificados vários sub-processos:

- *Submissão de uma declaração*. O processamento de uma declaração começa com a submissão e entrega do formulário da declaração. Isto tem lugar antes

– ou o mais tardar – no momento em que os bens se tornam disponíveis para verificação física. O formulário da declaração contém todos os dados que determinam como a declaração irá proceder ao longo do processo. Estas características são gravadas como atributos e casos e tomam um papel muito importante no encaminhamento do caso ao longo do processo. O sub-processo é adequado para ambas as declarações, electrónicas e escritas, assim como para a versão inicial e as novas versões.

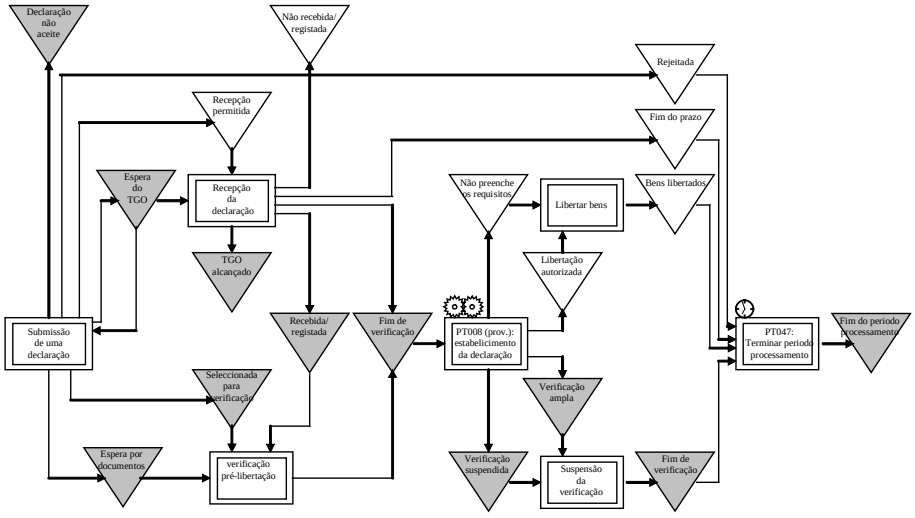


Figura 7.3. Exemplo do processo principal do Sagitta 2000

- *Aceitação da declaração*. O sub-processo de aceitação de uma declaração começa assim que a declaração é submetida. Este é um procedimento explícito, devido às implicações legais da aceitação de uma declaração alfandegária no CAC. O sub-processo espera até os bens estarem fisicamente presentes, depois o Sagitta 2000 estabelece o estado de "aceite". Mesmo depois da declaração ter sido aceite, a sua correcção e cancelamento pela parte declaradora ainda é possível. Estes são exemplos de eventos externos ao Serviço Alfandegário aos quais o serviço tem de responder.
- *Verificação de pré-lançamento*. O processo de verificação ou controlo acontece em paralelo com o processo de aceitação e é na teoria condicional. O processo de verificação é activado apenas quando é decidido que tem de haver uma inspecção (física). A eficácia da verificação é determinada usando os perfis de selecção contidos no sub-processo "submissão de declaração". Usando regras de decisão, a declaração é encaminhada para o sub-processo "verificação de pré-lançamento" ou então salta esse sub-processo.
- *Libertação de bens*. A libertação de bens pode acontecer assim que a declaração tenha sido aceite e qualquer verificação tenha ocorrido. PT008 é uma tarefa totalmente automática que liberta os bens. A libertação indica à parte declaradora que está autorizada para remover os bens correspondentes à declaração.

- *Suspensão da verificação*. Pode ser visto como um estado do Sagitta 2000 em que os bens podem, em princípio, ser libertados, mas a verificação não aconteceu ou não pôde ser completada. Em teoria, a suspensão da verificação ocorre independentemente da libertação de bens. Assim sendo, é modelada em paralelo com a tarefa de libertação. Uma vez a verificação completa e os bens libertados, o processo termina. Isto é feito executando a tarefa PT047.
- Um caso – i.e. uma declaração – eventualmente termina numa das seguintes condições: *Declaração recusada, Não recebida/registada*, ou *Fim do período de processamento*.

Submissão de uma declaração

A figura 7.4 mostra o sub-processo "submissão de uma declaração". Mais uma vez, é de salientar que esta não é a versão mais recente do processo: o Sagitta é um projecto ainda não finalizado. Podemos ver como um número de condições do mais alto nível do procedimento são repetidas. Estes fazem a ligação com o resto do processo ao mais alto nível.

O sub-processo é desenhado para verificar as declarações (*PT000* e *PT001*) e as novas versões das declarações (*PT039* e *PT040*) com base no seu conteúdo e – se estão em ordem – para registá-las (*PT007*). As declarações podem ser submetidas tanto electronicamente (*PT001*, *PT040a*, e *PT040b*) como por escrito (*PT000*, *PT039a*, e *PT039b*). O conteúdo das tarefas *PT039a* e *PT039b* é o mesmo: em ambos os casos, significa a correcção de uma declaração escrita. A *PT039a* é executada quando os bens aos quais a declaração pertence ainda não estão disponíveis (ou seja, quando ainda há um testemunho em *Espera pelo TGO*); por oposição, a *PT039b* é executada. Algumas vezes são necessárias verificações que o sistema não consegue resolver automaticamente. Pode ser necessário, por exemplo, envolver um dos contactos externos dos Serviços Alfandegário na verificação da declaração antes desta ser aceite pelo sistema. Outro exemplo de verificação de declaração que o sistema não pode executar é o de verificar se o lançamento de uma permissão num procedimento simplificado é permissível. Na figura 7.3, a *PT002a* corresponde à avaliação de uma nova declaração. Esta tarefa determina se o procedimento simplificado é permissível. A *PT002b* e a *PT002c* correspondem a uma avaliação semelhante de uma declaração corrigida.

Foi decidido introduzir seis tarefas separadas para a entrada de novas declarações electrónicas e escritas e subsequentes versões das declarações, por um lado, porque as tarefas para submeter uma declaração escrita têm um disparo diferente das submetidas electronicamente, por outro lado, no processo de negócio desejamos diferenciar explicitamente entre novas declarações, novas versões das declarações aceites e novas versões das declarações ainda não aceites. Para muitas declarações, não é necessário executar as verificações (adicionais) não automáticas entre as condições "organizações externas a informar" e "possível aceitação" (estas tarefas não são mostradas no exemplo). Para estas declarações, as tarefas *PT000*, *PT001*, *PT039a*, *PT039b*, *PT040a* e *PT040b* podem ser directamente seguidas pela *PT007*. Então, os pares de tarefas (*PT000-PT007*, *PT001-PT007*, *PT039a-PT007*, *PT039b-PT007*, *PT040a-PT007* e *PT040b-PT007*) podiam ser incorporados numa única tarefa (com a *PT007* como subtarefa opcional). Assim, é claramente feita uma decisão de modelação, por um lado, com o princípio de agregação de tempo, local e operação, por outro, a

associação das funcionalidades para uma única tarefa (*PT007*), sendo comparadas estas duas alternativas. Além disso, a solução escolhida tem a vantagem de todas as declarações passarem pela *PT007* e daqui serem encaminhadas apropriadamente. Então, a *PT007* actua como tarefa encaminhadora que aumenta a legibilidade do processo de negócio.

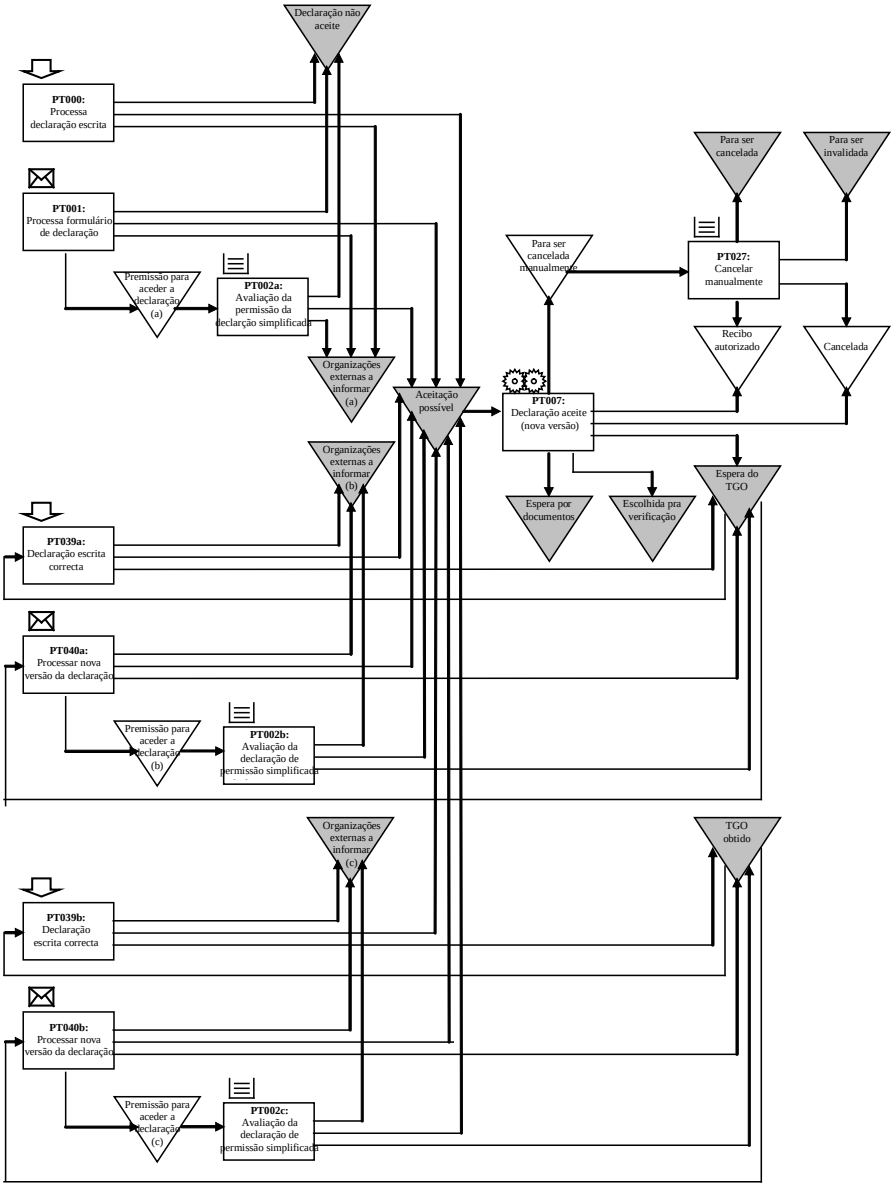


Figura 7.4. Exemplo da submissão de uma declaração

As figuras 7.3 e 7.4 apenas mostram uma parte de todo o processo. O processo de declaração contém mais de cinquenta tarefas individuais (no total, o Sagitta 2000 irá suportar mais de duzentas tarefas). Para cada uma destas, existe uma regra de decisão que determina o seu comportamento preciso. A figura 7.5 mostra as regras de decisão para algumas tarefas.

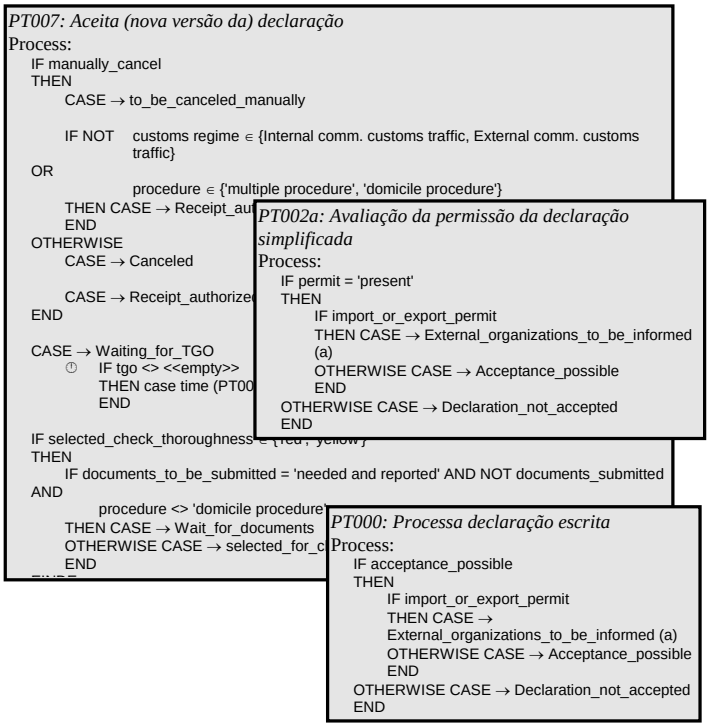


Figura 7.5. Regras de decisão para as tarefas PT000, PT002a e PT007

7.5 Execução dos Workflows num Sistema de Gestão de Workflow

O objectivo final consiste em incorporar os processos de negócio do Sagitta 2000 num sistema de gestão de *workflow*. Embora existam outras soluções técnicas em que o controlo do processo de negócio é nitidamente separado da aplicação, é política da Autoridade de Impostos Holandesa implementar *software standard* sempre que for possível. Por esta razão, foi previamente escolhido um produto de gestão de *workflow* para testar se a incorporação do Sagitta 2000 podia ser possível. Este produto de *workflow*, além de providenciar as funcionalidades necessárias para lidar com os complexos processos de negócio do Serviço Alfandegário, tem também de suportar as exigências da infra-estrutura técnica estabelecidas pela Autoridade dos Impostos e tem ainda de ser uma solução suficientemente sólida e robusta para funcionar com um elevado número de declarações e com os altos padrões exigidos pelo Serviço Alfandegário em termos de integridade e temporização.

7.5.1 Selecção de um sistema de gestão de workflow

Na selecção de um produto de *workflow*, a questão principal é se este irá permitir a execução do processo de negócio do Serviço Alfandegário. Foi feito um estudo de vários sistemas gestores de *workflow* para encontrar um produto que satisfaça um conjunto de exigências que inclui os seguintes requisitos:

- tem de ser possível modelar explicitamente os estados dos processos de negócio no sistema de gestão de *workflow*;
- têm de ser suportadas todas as formas de encaminhamento;
- têm de ser suportadas várias formas de disparos;
- tem de ser possível especificar uma hierarquia de processos de negócio;
- tem de ser possível extrair um caso do sistema de gestão de *workflow* e transferi-lo para outro sistema de gestão de *workflow* (funcionalidade exportar/importar); e
- tem de haver um suporte suficiente para os atributos de casos e regras de decisão.

Adicionalmente, foi feita uma breve pesquisa sobre os requisitos que o produto deveria satisfazer no que respeitava à distribuição de trabalho e à gestão da carga laboral para que estes aspectos pudessem ser tomados em conta durante a selecção. Os aspectos abordados incluíram regras de distribuição de trabalho, separação de funções e condições de autorização, oportunidades para o processamento em cadeia, em lotes, e outros.

A melhor forma de testar os aspectos listados anteriormente consiste em executá-los ao longo de uma parte do processo de negócio e regras de associação, com a ajuda de um perito do produto a ser avaliado. Isto irá rapidamente tornar claro se o produto fornece uma boa solução para a modelação dos estados, as várias formas de disparos, o método de associação desejado, a complexidade das regras de decisão, e outros. Em muitos dos produtos de *workflow*, tornou-se necessário traduzir as redes de Petri do Sagitta 2000 para uma linguagem do próprio produto antes que o processo fosse posto em produção. Durante esta tradução nem sempre foi possível encontrar, na linguagem do produto, uma solução adequada para todas as construções usadas no processo de origem.

Em 1998, o produto COSA (ver capítulo 5) foi o sistema de gestão de *workflow* escolhido para o Sagitta 2000. A decisão de escolher o COSA como produto de *workflow standard* para a Autoridade de Impostos Holandesa foi um resultado de uma proposta europeia. O COSA foi usado em vários projectos-piloto dentro da autoridade de Impostos Holandesa. Contudo, para o Sagitta 2000, o COSA não foi usado até este momento (Julho de 2000). Um projecto-piloto foi conduzido usando *software* personalizado e focando-se num pequeno fragmento de todo o processo (envolvendo cerca de dez tarefas).

7.5.2 Aspectos de distribuição

O Sagitta 2000 é um sistema distribuído – as suas estações de trabalho estão espalhadas por dezenas de postos alfandegários. O sistema consiste num *hub*

central e num conjunto de elementos locais. O *hub* coordena todo o sistema, e é também o lugar onde são executadas várias tarefas não interactivas. As tarefas interactivas são executadas pelos oficiais das alfândegas locais. A distribuição de funcionários é feita localmente, em cada posto alfandegário. A separação gestão/aplicação e central/local resulta numa estrutura em quatro partes ilustrada na figura 7.6.

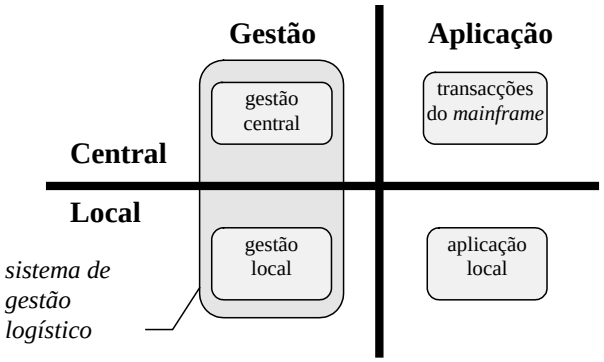


Figura 7.6. Divisão entre central e local e entre gestão e aplicação

O ambiente *mainframe* é usado centralmente. Localmente, a arquitectura usada é baseada no modelo cliente/servidor. Em 1998, o sistema de gestão de *workflow* COSA foi escolhido para a gestão local. Neste momento, não é definitivo se o COSA vai ser usado para os sistemas locais. Para o *hub* central as coisas são ainda mais complicadas, porque não existem sistemas de gestão de *workflow* disponíveis para o *mainframe* usado pela Autoridade dos Impostos. Além disso, são necessários altos níveis de desempenho e fiabilidade para o hub central. Ainda não é claro como será realizada a gestão da central. A Autoridade dos Impostos está a experimentar vários protótipos de sistemas de gestão (e.g. sistemas de controlo de fluxo). Estes protótipos são baseados em redes de Petri e nas técnicas de modelação apresentadas neste livro.

O ponto de partida para o sistema de gestão do Sagitta 2000 é baseado no princípio de que um caso (i.e. a declaração alfandegária) está apenas num só local em determinado instante. Por isso, o caso está no *hub* central ou num posto local e não pode ser trabalhado simultaneamente nos dois locais. Algumas vezes é necessário transferir um caso de um lugar para outro – isto é, de um posto local para outro ou do *hub* central para um posto local. Conceptualmente, podemos comparar a transferência de um caso com a remoção de todos os testemunhos pertencentes a uma declaração de um processo e colocando-os noutro. Ao transferir um caso, a definição do estado é também crucial. Muitas vezes, os sistemas gestores de *workflow* que não são baseados em redes de Petri abstraem-se do estado, sendo assim inadequados nesse aspecto.

7.5.3 Mapeamento do processo para o WFMS

Embora o sistema de gestão de *workflow* escolhido (COSA) suporte redes de Petri, não é possível transferir processos. Isso acontece porque o sistema

escolhido não aceita todas as construções permitidas numa rede de Petri de alto nível. Por isso, é necessário desenvolver soluções *standard* que não diminuam a funcionalidade desejada. Todas estas soluções foram descritas no manual de execução, que é seguido de forma precisa durante a execução, logo as diferenças entre as redes de Petri de alto nível e a linguagem usada pelo sistema de *workflow* são sempre resolvidas de um modo consistente. Alguns exemplos incluídos no manual de execução são:

- a forma como os atributos devem ser tratados e os nomes dados a estes atributos;
- a forma como as regras de decisão são estabelecidas;
- a forma como os processos automáticos são estabelecidos;
- a forma como os disparos baseados no tempo são tratados;
- a inspecção de uma condição por uma tarefa, ou a execução de uma iteração (inspecção de um caso e seu retorno à mesma condição);
- a criação de um caso como resultado de uma mensagem exterior; e
- a execução de disparos entre dois processos.

A figura 7.7 mostra uma pequena parte da definição de um dos processos do Sagitta 2000. Para execução no COSA, as regras de decisão são traduzidas em condições para os arcos entre as transições e os lugares. Como ilustrado, estas condições podem tornar-se extremamente complexas.

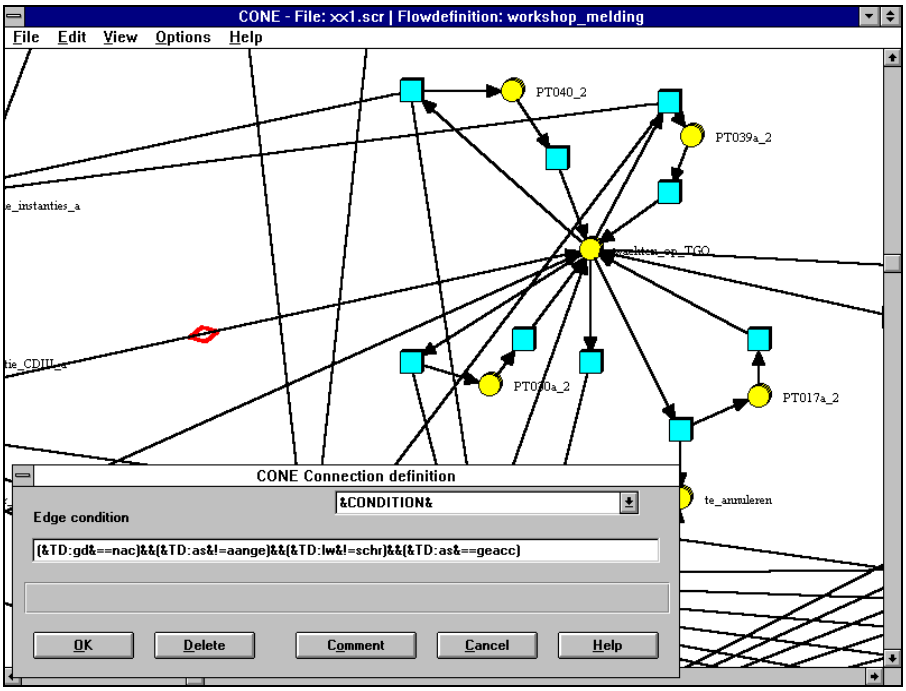


Figura 7.7. Parte da definição de processo no COSA

Tal como é requerido pela metodologia IPSD descrita no capítulo anterior, os *workshops* interactivos tiveram um papel crucial na validação dos processos de negócio. Estes *workshops* estimularam e suportaram os peritos alfandegários a efectuar testes cuidadosos aos processos de negócio durante as simulações dos processos usando o sistema de gestão de *workflow*.

De notar que a figura 7.7 só foi adicionada com propósitos de ilustração. O COSA foi seleccionado como sistema de *workflow* em 1998. Contudo neste momento não é claro se o COSA irá na verdade ser usado como base para o Sagitta. A primeira versão de produção do Sagitta, suportando apenas uma pequena parte do processo total, irá usar *software* personalizado.

7.6 Algumas Lições até ao Momento…

Embora tenham sido previamente procuradas oportunidades para por em prática ideias associadas à gestão de *workflows* nas várias secções da Autoridade de Impostos Holandesa, o Sagitta 2000 é o primeiro projecto cujo desenho teve sucesso na separação das aplicações e da logística (gestão). Para atingir isto, foi feita uma investigação extensiva nos (tipos de) blocos de construção a partir dos quais o sistema foi construído e em métodos para modelar e especificar os processos de negócio. Eventualmente, foi escolhida uma metodologia baseada em redes de Petri. Alguns aspectos relacionados com metodologias e arquitecturas foram testados num projecto denominado de praticabilidade. Durante este projecto-piloto, os processos de negócio foram incorporados no sistema de gestão de *workflow* seleccionado.

As lições mais importantes até ao momento foram as seguintes:

- Uma forma adequada para especificar os procedimentos do Serviço Alfandegário consiste em usar redes de Petri. Não foram identificadas situações nas quais as redes de Petri não tenham sido capazes de modelar um procedimento desejado.
- As redes de Petri são também, em princípio, bem compreendidas pelos peritos do Serviço Alfandegário. A representação explícita dos estados de um caso contribui para uma melhor compreensão do *workflow* a ser modelado.
- É muito importante que a equipa de arquitectos (técnicos de informação) de processos e os peritos alfandegários trabalhem bem em conjunto.
- Uma abordagem formal é a melhor opção para descrever os processos de negócio e incorporá-los num sistema de gestão de *workflow*. No Sagitta 2000, foram realizados *workshops* para que (outros) peritos alfandegários testassem os processos de negócio. Ao invocar uma aplicação *standard* por cada tarefa que mostra ao utilizador uma descrição textual das responsabilidades da tarefa, em vez de executar a própria aplicação da tarefa, não é necessário às tarefas de aplicação serem construídas antes do processo de negócio ser testado.
- Até agora, foi dada pouca atenção aos requisitos funcionais no que diz respeito à distribuição de trabalho e a gestão da carga laboral. Foi feita uma pesquisa a estes aspectos. As experiências iniciais mostram que as organizações ainda consideram difícil apreciar as oportunidades criadas pela gestão de *workflow*. É necessário dar uma especial atenção à formulação da primeira versão dos requisitos. Esta versão faz um uso limitado das

oportunidades dadas pelos sistemas de gestão de *workflow*. Actualmente, uma pesquisa considerável sobre as oportunidades e as novas potencialidades oferecidas pelos sistemas de *workflow* está em curso. Esta pesquisa aborda os seguintes aspectos:

(1) Autorização dos utilizadores para efectuar tarefas: competências do utilizador e separação de funções;
(2) Gestão da carga laboral: distribuição eficiente do trabalho pelos funcionários disponíveis; e
(3) Assegurar que o processo progride, e que avisos são emitidos quando ocorre uma estagnação.

- Actualmente, as maiores "dores de cabeça" são causadas por problemas tecnológicos. A introdução de sistemas de gestão de *workflow* num ambiente que fixa altos padrões para a sua infra-estrutura técnica requer que seja dada grande atenção à tecnologia. O Serviço Alfandegário exige a disponibilidade ininterrupta de certos sub-processos, um alto nível de robustez e uma completa integridade do sistema e das suas bases de dados. Estes altos padrões tornam difícil a introdução da tecnologia de *workflow*.

Conclusão

Com o Sagitta 2000, foi realizada uma melhoria consistente e coerente da forma como os sistemas podem ser criados e como a gestão de *workflows* pode ser integrada num novo sistema de informação. Porém, este trabalho está longe do fim: em ambas as frentes técnicas e organizacionais, existem ainda muitos obstáculos a ultrapassar. Não obstante, existe a confiança que isto será realizado e as expectativas dentro da organização são elevadas. Um pouco à parte do aspecto da gestão de *workflow*, o Sagitta 2000 já provou ser muito proveitoso, reconsiderando completamente e definindo explicitamente os processos de negócio do Serviço Alfandegário. O novo processo tenta resolver as ineficiências e inconsistências dos existentes, e cumpre os mais recentes requisitos dos processos de negócio impostos pelo Serviço Alfandegário (CAC e o Conceito de Cliente).

EXERCÍCIOS

Exercício 7.1 Viajar pela Universidade Algures

Aplique a técnica de modelação descrita neste livro ao processo de *workflow* da seguinte agência de viagens.

Há algum tempo atrás, a reitoria da Universidade Algures (UA) decidiu abrir uma agência de viagens no *campus*. Supõe-se que a nova agência organize, tanto, viagens de negócio como viagens privadas para funcionários da UA. Porém, o serviço não é como a reitoria esperava. A principal reclamação é que tanto a organização de uma viagem como o seu ajuste financeiro demoram demasiado tempo. Então, a reitoria começou uma investigação. Entrevistas com

várias pessoas envolvidas providenciaram a seguinte descrição do processo. (Para evitar confusões entre funcionários da UA que querem marcar uma viagem e funcionários que estão envolvidos na organização da viagem, os primeiros são chamados de clientes.)

Todo o processo começa quando alguém entra na agência de viagens para marcar uma viagem. Um funcionário da agência regista toda a informação relevante do cliente. A agência mantém um ficheiro separado para cada viagem. Um assunto importante é se o cliente quer marcar uma viagem privada, de negócios ou uma combinação de ambas. Aproximadamente vinte por cento das viagens organizadas pela agência são privadas.

As viagens privadas são fáceis. A agência tem um funcionário dedicado à organização das viagens privadas. Assim que os desejos de um cliente são registados, este pode começar com a organização da viagem.

As viagens de negócios são mais complicadas. A agência tem dois funcionários para a organização das viagens de negócios (embora um deles trabalhe apenas três dias por semana). Para cada viagem, existe sempre um único funcionário responsável que executa o maior número possível de tarefas para uma viagem. Desta forma, o serviço aos clientes deve ficar garantido. Para as viagens de negócios, o cliente precisa de uma licença. Usualmente, os clientes familiarizados com o processo já vêm com a licença preenchida. Aos clientes que chegam sem a licença é fornecida uma licença em branco que podem preencher mais tarde e que depois têm de devolver à agência. As licenças de viagem são sempre verificadas antes de se tomar qualquer outra acção. Se uma licença não está preenchida correctamente é devolvida ao cliente com o pedido de fornecer a informação em falta e enviar a licença de volta o mais cedo possível. No caso de uma licença não ser devolvida a tempo, a agência de viagens deixa de poder garantir a organização da viagem dentro do tempo previsto. Nestas raras ocasiões, é enviada uma notificação ao cliente e o ficheiro é encerrado. Se a licença de viagem estiver correcta, a organização da viagem pode começar. Porém, primeiro é enviada uma cópia do ficheiro para o departamento de finanças da UA, porque este departamento é responsável pelo aspecto financeiro da viagem.

Um funcionário do departamento de finanças da UA verifica se o cliente tem permissão para efectuar viagens de negócios pagas pela UA. Os resultados desta verificação são enviados para a agência de viagens através de um memorando interno. Se o resultado for negativo para o cliente, o departamento de finanças não faz qualquer pagamento, o que é um caso raro porque os clientes usualmente sabem quando têm permissão para fazer viagens de negócios. Se o resultado for positivo, o departamento de finanças efectua um pagamento na conta bancária do cliente. Além disso, também paga qualquer taxa de registo que pode ser necessária em caso de visitas a conferências. Finalmente, paga os voos das viagens feitas com uma finalidade de negócio. Porém, estes pagamentos apenas podem ser feitos após o departamento de finanças ter recebido da agência de viagens a informação detalhada dos preços. Depois de efectuados todos os pagamentos necessários, o departamento de finanças deixa de estar envolvido na organização da viagem. Contudo, depois do regresso do cliente, o departamento de finanças controla a declaração do cliente (ver abaixo).

Para preparar uma viagem (privada ou de negócios), a agência de viagens começa sempre pela reserva dos voos. Se uma viagem envolve um ou mais

voos, o funcionário responsável começa por preparar um horário de voo que inclua as horas de partida e chegada de todos os voos assim como informações sobre os preços. Depois, o cliente é chamado para aprovar o horário. Se o cliente não aprovar o horário, é preparada uma nova proposta e o cliente é contactado novamente. Quando um cliente aprova o horário, é marcada uma reunião com o cliente para o pagamento em dinheiro ou com cartão de crédito. No caso da viagem ser (parcialmente) de negócios, a agência de viagens tem de esperar pelo memorando do departamento de finanças que indica se o cliente tem ou não permissão para fazer viagens de negócios pela UA. Se o memorando é negativo, o funcionário da agência de viagens responsável pela viagem contacta o cliente para explicar o problema. Se o cliente ainda quiser efectuar a viagem, ele ou ela têm de pagar todos os custos e é marcada uma reunião para pagar os voos. Muitas vezes, o cliente decide cancelar a viagem, neste caso o ficheiro é fechado. Se o memorando for positivo, a agência de viagens determina os custos dos voos de negócios e, se aplicável, os custos dos voos privados. A informação relevante dos voos de negócios é enviada para o departamento de finanças, que lida com o pagamento. No caso dos voos privados, o cliente é contactado para marcar uma reunião para tratar do pagamento.

O memorando interno que a agência de viagens recebe do departamento de finanças é também usado para determinar se um pedido tem de ser enviado para o escritório do banco local (situado no *campus* perto da agência de viagens) para preparar o dinheiro e os cheques de viagem para o cliente. Tal pedido é feito sempre que uma viagem de negócios é permitida (no caso das viagens privadas, o cliente tem que tratar de adquirir o dinheiro e cheques por ele próprio.)

A tarefa do banco no processo é muito directa. Quando recebe um pedido, um funcionário do banco prepara o dinheiro e os cheques de viagem e envia-os para a agência de viagens. Se um cliente devolver dinheiro e/ou cheques depois da viagem, a informação do valor exacto que é usado pelo cliente é enviada para o departamento de finanças. O departamento de finanças necessita dessa informação para processar a declaração do cliente. No caso de um cliente não devolver o dinheiro ou cheque a tempo, o montante supostamente gasto pelo cliente é fixado no valor do dinheiro e cheques entregues ao cliente antes da viagem.

Depois de ajustes nos voos e qualquer voo privado terem sido pagos, o funcionário responsável reserva os hotéis e transportes locais (comboio, carro, etc.). Ele também imprime *vouchers* para qualquer hotel reservado. Quando o dinheiro e os cheques são recebidos do banco e são recebidos todos os bilhetes de avião do escritório central da agência de viagens em Algures, onde são impressos, o funcionário coloca todos os documentos numa pasta para o cliente. A agência tem de ter certeza que tudo está pronto pelo menos três dias úteis antes da viagem começar porque, depois, o cliente levanta os documentos. Nesse ponto, o envolvimento da agência com a viagem está terminado. No caso de uma viagem privada, isto também significa que o processo está completo. Enquanto que no caso de ser uma viagem de negócios, a declaração do cliente ainda necessita ser processada.

Como já foi mencionado, o departamento de finanças toma conta do processamento de declarações. Quando recebe uma declaração de um cliente e a informação necessária do banco, um funcionário do departamento de finanças

processa a declaração e calcula o saldo. O resultado tem de ser aprovado pelo director do departamento de finanças. No caso de ocorrer um erro, o funcionário tem de fazer as correcções necessárias.

Depois da declaração ter sido aprovada pelo director, o saldo é estabelecido no próximo pagamento de salário ao cliente. Além disso, o custo total da viagem é deduzido do orçamento da faculdade ou de outra unidade onde o cliente trabalha. Se um cliente não entregar a sua declaração a tempo (até um mês após o final da viagem), o departamento de finanças assume que o custo total da viagem iguala a soma do pagamento adiantado e o valor do dinheiro e dos cheques dados ao cliente.

A reitoria da UA pensa que a principal razão pela qual este processo demora tanto tempo é porque a coordenação entre os três departamentos envolvidos é reduzida e pobre e acredita ainda que um sistema de *workflow* pode providenciar uma solução. Como ponto de partida, irão querer receber um relatório cobrindo os seguintes assuntos.

(a) Classificação de todos os recursos envolvidos no processo actual, distinguindo papéis e grupos.
(b) Um modelo do processo da situação actual, incluído informação sobre papéis, grupos e disparos.
(c) Uma análise da classificação de recursos e do modelo de processo, usando as directivas para o (re)desenho de processos discutidas nos capítulos anteriores.
(d) Um modelo melhorado da classificação/processo de recursos, baseado nos resultados da análise.

(Página deixada propositadamente em branco)

APÊNDICE A.

TEORIA DOS WORKFLOWS

Este livro oferece técnicas concretas e guias para desenhar processos de *workflow* complexos. Embora a necessidade de uma base teórica fosse realçada, definições formais e notações têm sido evitadas ao máximo possível para facilitar a leitura. Este apêndice introduz as bases teóricas para as técnicas de modelação usadas ao longo deste livro.

A situação actual, no que diz respeito ao *software* de gestão de *workflow*, é comparável com a situação relativa ao *software* de gestão de bases de dados no início dos anos 70. Neste período, muitos dos pioneiros no campo da gestão de sistemas de gestão de bases de dados (*DataBase Management Systems - DBMSs*) usaram os seus próprios conceitos *ad hoc*. Esta situação de desordem e falta de consenso resultou num cenário incompreensível de DBMSs. No entanto, surgiram *standards*, tal como o modelo de dados relacional e o modelo de entidade–relacionamento que levaram a uma base formal comum para muitos DBMSs. Como resultado, o uso dos DBMSs aumentou. Há muitas semelhanças entre os sistemas de gestão de *workflow* (WFMSs) e os DBMSs do início dos anos 70. Apesar dos esforços da *Workflow Managment Coalition*, ainda falta um *standard* conceptual real. Como resultado, muitas organizações estão relutantes em usar o *software* de gestão de *workflow* existente.

O modelo relacional de dados e o modelo de entidade–relacionamento serviu como um catalisador para o uso e funcionalidade dos DBMSs. Modelos comparáveis estão em falta nos WFMSs. Um WFMS foca várias perspectivas e é utópico assumir que um modelo elementar comparável ao modelo de dados relacional ou ao modelo entidade–relacionamento possa capturar todos os aspectos relevantes. Contudo, para a perspectiva mais dominante, que é a perspectiva do processo (fluxo de controlo), parece haver um consenso nos principais conceitos. Na nossa opinião, as redes de Petri constituem uma boa base para a normalização desta perspectiva. Inspirados por experiências práticas, apercebemo-nos que muitas das características do formalismo das redes de Petri são necessárias no contexto de gestão de *workflow*.

No capítulo 2 deste livro motivamos o leitor para o uso das redes de Petri como linguagem de desenho. Na nossa opinião, as redes de Petri constituem um bom ponto de partida para uma *teoria de workflow*. Neste apêndice centramo-nos na base desta teoria. Primeiro, introduzimos o formalismo associado às redes de Petri. Depois, formalizamos a noção de correcção usada no capítulo 4 (i.e., *soundness*). Finalmente, demonstramos que a teoria das redes de Petri pode ajudar a encontrar caracterizações estruturais (i.e., padrões de desenho) de correcção e técnicas de análise eficientes.

A.1 Redes de Petri

Esta secção introduz a terminologia básica e notação das redes de Petri. Os leitores familiarizados com redes de Petri podem optar por passar à secção seguinte.

Uma rede de Petri clássica é um grafo direccionado bipartido com dois tipos de nós chamados de *lugares* e *transições*. Os nós são interligados usando *arcos* direccionados. As ligações entre dois nós do mesmo tipo não são permitidas e os lugares são representados por círculos e as transições por rectângulos.

DEFINIÇÃO 1 (Rede de Petri). Uma rede de Petri é um triplo (P, T, F):

- P é um conjunto finito de lugares;
- T é um conjunto finito de transições ($P \cap T = \varnothing$); e
- $F \subseteq (P \times T) \cup (T \times P)$ é um conjunto de arcos (relação de fluxo).

Um lugar p é denominado de *lugar de entrada* de uma transição t sse (se e só se) existir um arco directo de p para t. Um lugar p é chamado de *lugar de saída* de uma transição t sse existir um arco directo de t para p. Usamos a notação $\bullet t$ para denotar o conjunto de lugares de entrada para uma transição t. As notações $t\bullet$, $\bullet p$ e $p\bullet$ têm significados semelhantes, ou seja, $p\bullet$ é o conjunto de transições que partilham p como lugar de entrada. De notar que não consideramos arcos múltiplos de um nó para outro.

Em qualquer altura, um lugar contém zero ou mais *testemunhos*, desenhados usando pontos pretos. Um *estado* M, muitas vezes referido por marcação, corresponde à distribuição de testemunhos pelos lugares, ou seja, $M \in P \rightarrow \text{IN}$. Iremos representar um estado da seguinte forma: $1p_1+2p_2+1p_3+0p_4$ é o estado com um testemunho no lugar p_1, dois testemunhos em p_2, um testemunho em p_3 e nenhum testemunho em p_4. Podemos também representar este estado da seguinte forma: $p_1+2p_2+p_3$. Para comparar estados definimos uma ordem parcial entre eles. Para quaisquer dois estados M_1 e M_2, $M_1 \leq M_2$, sse para todo o $p \in P$: $M_1(p) \leq M_2(p)$, onde $M(p)$ denota o número de testemunhos no lugar p no estado M.

O número de testemunhos pode variar durante a execução da rede. As transições são os componentes activos numa rede de Petri: elas mudam o estado da rede de acordo com a seguinte *regra de disparo*:

(1) A transição t é considerada *pronta* sse cada lugar de entrada p de t contém pelo menos um testemunho.
(2) Uma transição *pronta* pode *disparar*. Se a transição t dispara, então t *consome* um *testemunho* de cada lugar de entrada p de t e *produz* um testemunho para cada lugar de saída p de t.

Dada uma rede de Petri (P, T, F) e um estado M_1, temos as seguintes notações:

- $M_1 \xrightarrow{t} M_2$: a transição t está pronta no estado M_1 e o disparo de t em M_1 resulta no estado M_2.
- $M_1 \rightarrow M_2$: existe uma transição t tal que $M_1 \xrightarrow{t} M_2$.

- $M_1 \xrightarrow{\sigma} M_n$: A sequência de disparos $\sigma = t_1 t_2 t_3 \ldots t_{n-1}$ leva do estado M_1 para um estado M_n através de um conjunto (possivelmente vazio) de estados intermédios $M_2, \ldots, M_{n-1}$, i.e., $M_1 \xrightarrow{t1} M_2 \xrightarrow{t2} \ldots \xrightarrow{tn-1} M_n$.

Um estado M_n é chamado *alcançável* a partir de M_1 (notação $M_1 \xrightarrow{*} M_n$) sse existir uma sequência de disparos σ tal que $M_1 \xrightarrow{\sigma} M_n$. Note que a sequência de disparo vazia é também permitida, i.e., $M_1 \xrightarrow{*} M_1$.

Usamos (PN, M) para denotar a rede de Petri PN com um estado inicial M. Um estado M' é um *estado alcançável* de (PN, M) sse $M \xrightarrow{*} M'$.

Vamos definir algumas propriedades para as redes de Petri. Primeiro, vamos definir propriedades relacionadas com a dinâmica das redes de Petri e, depois, vamos descrever algumas propriedades estruturais.

DEFINIÇÃO 2 (Viva). Uma rede de Petri (PN, M) é viva sse para cada estado alcançável M' e para cada transição t existe um estado M'' alcançável de M' que torna a transição t pronta.

Uma rede de Petri é *estruturalmente viva* se existir um estado inicial tal que a rede seja viva.

DEFINIÇÃO 3 (Limitada, Segura). Uma rede de Petri (PN, M) é limitada sse para cada lugar existe um número natural n tal que para cada estado alcançável o número de testemunhos em p é menor que n. A rede é segura sse para cada lugar o número máximo de testemunhos não exceder 1.

Uma rede de Petri é *estruturalmente limitada* se a rede é limitada para qualquer estado inicial.

DEFINIÇÃO 4 (Bem formada). Uma rede de Petri PN é bem formada sse existe um estado M tal que (PN, M) é viva e limitada.

Os caminhos ligam os nós através de uma sequência de arcos.

DEFINIÇÃO 5 (Caminho, Elementar, Sem conflito). Seja PN uma rede de Petri. Um caminho C de um nó n_1 para um nó n_k é uma sequência $(n_1, n_2, \ldots, n_k)$ tal que $(n_i, n_{i+1}) \in F$ para $1 \leq i \leq k\text{-}1$. C é elementar sse, para quaisquer dois nós n_i e n_j em C, $i \neq j \Rightarrow n_i \neq n_j$. C é sem conflito sse para qualquer lugar n_j em C e qualquer transição n_i em C, $j \neq i\text{-}1 \Rightarrow n_j \notin {}^\bullet n_i$.

Por conveniência, introduzimos o operador alfabético α nos caminhos. Se $C = (n_1, n_2, \ldots, n_k)$, então $\alpha(C) = \{n_1, n_2, \ldots, n_k\}$.

DEFINIÇÃO 6 (Fortemente ligada). Uma rede de Petri é fortemente ligada sse, para todo o par de nós (i.e., lugares e transições) x e y, existe um caminho de x para y.

DEFINIÇÃO 7 (Livre escolha). Uma rede de Petri é de livre escolha sse, para todo o par de transições t_1 e t_2, ${}^\bullet t_1 \cap {}^\bullet t_2 \neq \varnothing$ implica que ${}^\bullet t_1 = {}^\bullet t_2$.

DEFINIÇÃO 8 (Máquina de estados). Uma rede de Petri é uma máquina de estados sse cada transição tem, exactamente, um lugar de entrada e um lugar de saída.

DEFINIÇÃO 9 (Componente-S). Uma sub-rede $PN_s = (P_s, T_s, F_s)$ é chamada de Componente-S de uma rede de Petri $PN = (P, T, F)$ se $P_s \subseteq P$, $T_s \subseteq T$, $F_s \subseteq F$, PN_s é fortemente ligada, PN_s é uma máquina de estado, e para todo o $q \in P_s$ e $t \in T$: $(q, t) \in F \Rightarrow (q, t) \in F_s$ e $(t, q) \in F \Rightarrow (t, q) \in F_s$.

DEFINIÇÃO 10 (Cobertura-S). Uma rede de Petri possui Cobertura-S sse para qualquer nó existe um componente-S que contém esse nó.

Consulte as referências [9, 15] para uma introdução mais elaborada sobre estas notações. A noção de Cobertura-S está relacionada com as noções de lugares e transições invariantes [9, 14, 15]. Um *lugar invariante* atribui um peso a cada lugar, de tal forma que nenhuma transição pode mudar a "soma pesada dos testemunhos". A soma pesada dos testemunhos é definida como a soma de todos os testemunhos multiplicados pelos pesos dos lugares correspondentes: isto é, uma função w é um lugar invariante se para qualquer estado M_1 e qualquer transição t, se verifique,

$$M_1 \xrightarrow{t} M_2: \sum_{p \in P} w(M_1(p)) = \sum_{p \in P} w(M_2(p)) \cdot$$

Note-se que os lugares invariantes são estruturais, ou seja, não dependem de um estado inicial. Os lugares invariantes correspondem às leis de conservação. Um lugar invariante é *semipositivo* se não atribuir pesos negativos às transições. Os lugares invariantes positivos atribuem um peso positivo a cada lugar. Note-se que cada Componente-S corresponde a um lugar invariante semipositivo. Para além disso, se uma rede de Petri tem Cobertura-S, então existe um invariante positivo. As *transições invariantes* são a dualidade dos lugares invariantes. Uma transição atribui um peso a cada transição tal que se toda a transição disparar um número específico de vezes, o estado inicial é recuperado. Os pesos negativos correspondem a "disparos para trás". Uma rede de Petri que é viva e possui Cobertura-S (ou limitada) tem uma transição invariante positiva.

A.2. WF-Nets

A gestão de workflows tem muitas perspectivas. A perspectiva do processo (i.e., o controlo do fluxo) é a mais proeminente, porque o núcleo de qualquer sistema de *workflow* é formado pelos processos que suporta. Na dimensão de controlo de fluxo, blocos de construção, tais como o AND-split, AND-join, OR-split e OR-join são usados para modelar encaminhamentos sequenciais, condicionais, paralelos e iterativos. Claramente, uma rede de Petri pode ser usada para especificar o encaminhamento de casos. As *tarefas* são modeladas usando transições e as dependências causais são modeladas por lugares e arcos. De facto, um lugar corresponde a uma *condição* que pode ser usada como pré e/ou pós condição para as tarefas. Um AND-split corresponde a uma transição

com dois ou mais lugares de saída e um AND-join corresponde a uma transição com dois ou mais lugares de entrada. Os OR-splits/OR-joins correspondem a lugares com múltiplos arcos de saída e de entrada. Além disso, em [1] é mostrado que a utilização de redes de Petri também permite a construção de encaminhamentos úteis mas ausentes em muitos WFMSs.

Uma rede de Petri que modele a dimensão do controlo do fluxo de um *workflow* é chamada de *WorkFlow net (WF-net)*. Deve ser notado que uma WF-net especifica o comportamento dinâmico de um único caso isolado.

DEFINIÇÃO 11 (WF-net). Uma rede de Petri $PN = (P, T, F)$ é uma WF-net (Workflow net) sse:

(i) Existe um lugar de origem $i \in P$ tal que $\bullet i = \emptyset$;

(ii) existe um lugar final $o \in P$ tal que $o\bullet = \emptyset$; e

(iii) todo o nó $x \in P \cup T$ está num caminho de i para o.

Uma WF-net tem um lugar de entrada (i) e um lugar de saída (o) porque qualquer caso tratado pela WF-net é criado quando este entra no WFMS e é removido quando está completamente tratado pelo WFMS; por outras palavras, uma WF-net especifica o ciclo de vida de um caso. O terceiro requisito da definição 11 foi adicionado para evitar a existência de tarefas e/ou condições soltas, ou seja, tarefas e condições que não contribuem para o processamento de casos.

Dada a definição de uma WF-net é fácil derivar as seguintes propriedades.

PROPOSIÇÃO 1. (Propriedades de WF-nets). Seja $PN = (P, T, F)$ uma rede de Petri.

- Se PN é uma WF-net com um lugar de origem i, então para qualquer lugar $p \in P$: $\bullet p \neq \emptyset$ ou $p = i$, i.e., i é o único lugar de origem;
- Se PN é uma WF-net com lugar final o, então para qualquer lugar $p \in P$: $p\bullet \neq \emptyset$ ou $p = o$, i.e., o é o único lugar final;
- Se PN é uma WF-net e adicionamos uma transição t^* a PN, que liga o lugar final o ao lugar de origem i (i.e., $\bullet t^* = \{o\}$ e $t^*\bullet = \{i\}$), então a rede de Petri resultante é fortemente ligada;
- Se PN tem um lugar de origem i e um lugar final o e adicionar uma transição t^* que liga o lugar final o ao lugar de origem i então obtemos uma rede fortemente ligada, então todo o nó $x \in P \cup T$ está num caminho de i para o em PN e PN é uma WF-net.

A figura A.1 mostra uma WF-net que modela um processo de reclamação. Primeiro, a reclamação é registada (tarefa *registar*), então, em paralelo, um questionário é enviado para o queixoso (tarefa *enviar_questionário*) e a reclamação é avaliada (tarefa *avaliar*). Se o queixoso devolve o questionário no prazo de duas semanas, a tarefa *processar_questionário* é executada. Se o questionário não é devolvido dentro de duas semanas, o resultado do questionário é descartado (tarefa *time_out*). Baseado no resultado da avaliação, a queixa é processada ou não. O actual processamento da reclamação (tarefa *processar_reclamação*) é atrasado até que a condição *c5* seja satisfeita, ou seja, o questionário é processado ou ocorre um *time_out*. O processamento da reclamação é verificado através da tarefa *verificar_processamento*. Finalmente, a

tarefa *arquivar* é executada. Note-se que o encaminhamento sequencial, condicional, paralelo e iterativo estão presentes neste exemplo.

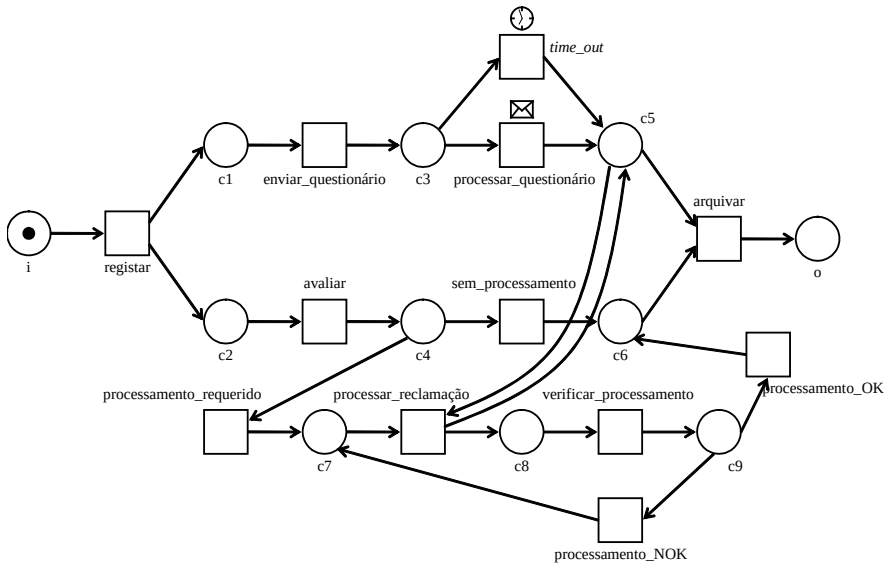


Figura A.1. Uma WF-net para o processamento de reclamações

A WF-net ilustrada na figura A.1 mostra claramente que nos centramos na dimensão do controlo de fluxo. Abstraímo-nos dos recursos, das aplicações e das plataformas tecnológicas. Adicionalmente, também nos abstraímos dos atributos dos casos e dos disparos. Os atributos dos casos são usados para resolver escolhas (OR-split); Por outras palavras, a escolha entre a tarefa *processamento_requerido* e a tarefa *sem_processamento* é (parcialmente) baseada no conjunto dos atributos dos casos durante a execução da tarefa avaliar. A escolha entre *processamento_OK* e *processamento_NOK* é resolvida testando os atributos do caso aferido pela tarefa *verificar_processamento*. Nas WF-nets abstraímo-nos dos atributos dos casos ao introduzir escolhas não-determinísticas na rede de Petri. Se não nos abstraímos desta informação, iremos ter de modelar o comportamento (desconhecido) das aplicações usadas em cada uma das tarefas e a análise tornar-se-ia intratável. Na figura A.1 indicamos que o *time_out* e a tarefa de *processar_questionário* requerem disparos. O símbolo do relógio denota um disparo de tempo e o símbolo do envelope denota um disparo externo. A tarefa *time_out* requer um disparo de tempo ("duas semanas passaram") e o *processar_questionário* requer um disparo com uma mensagem ("o questionário foi devolvido"). Um disparo pode ser visto como uma condição adicional que necessita de ser satisfeita. Neste capítulo não consideramos essas condições de disparos e assumimos que o ambiente se comporta de forma justa, ou seja, a vivacidade de uma transição não é impedida de ser disparada pela ausência de um disparo específico. Como resultado, todo o disparo irá eventualmente ocorrer.

A.3 Correcção

Nesta secção resumimos alguns dos resultados básicos para as WF-nets apresentados em [2]. O resto deste capítulo irá ser construído sobre esses resultados.

Os três requisitos enunciados na definição 11 podem ser verificados estaticamente. Por outras palavras, eles estão apenas relacionados com a estrutura da rede de Petri. Contudo, existe um outro requisito que deve ser satisfeito:

Para qualquer caso, o processo irá eventualmente terminar e, no momento em que termina, existe um testemunho no lugar *o* e todos os outros lugares estão vazios.

Para além disso, não devem existir tarefas mortas, deve ser possível executar uma tarefa arbitrária seguindo caminhos apropriados através da WF-net. Estes dois requisitos adicionais correspondem à chamada *propriedade de correcção*.

DEFINIÇÃO 12 (Correcto). Um processo modelado por uma WF-net $PN = (P, T, F)$ é correcto sse:

(i) Para todo o estado M alcançável a partir do estado i, existe uma sequência de disparos que conduz do estado M para o estado o. Formalmente:

$\forall_M (i \xrightarrow{*} M) \Rightarrow (M \xrightarrow{*} o)$;

(ii) O estado o é o único estado alcançável a partir do estado i com pelo menos um testemunho no lugar o. Formalmente:

$\forall_M (i \xrightarrow{*} M \wedge M \geq o) \Rightarrow (M = o)$;

(iii) Não existem transições mortas em (PN, i). Formalmente:

$\forall_{t \in T} \exists_{M,M'} i \xrightarrow{*} M \xrightarrow{t} M'$.

Note-se que a propriedade de correcção relaciona-se com a dinâmica de uma WF-net. O primeiro requisito da definição 12 afirma que começando de um estado inicial (estado i)[2], é sempre possível alcançar um estado com um testemunho no lugar o (estado o). Se assumimos uma noção de justiça forte, então o primeiro requisito implica que eventualmente o estado o é alcançado. Uma justiça forte significa que em cada sequência de disparo infinita, cada transição dispara infinitamente. A suposição de justiça é razoável no contexto da gestão de *workflow*: todas as escolhas são feitas (implicitamente ou explicitamente) por aplicações, humanos ou actores externos. Claramente, estes não devem introduzir um ciclo infinito. Repare-se que as notações tradicionais de justiça (i.e., formas fracas de justiça com apenas condições locais, ex., se uma transição está frequentemente pronta infinitamente, esta irá disparar eventualmente) não são suficientes. Consulte as referências [3, 13] para mais detalhes. O segundo requisito afirma que no momento em que um testemunho

é colocado no lugar *o*, todos os outros lugares devem estar vazios. Algumas vezes, o termo *terminação adequada* é usado para descrever os primeiros dois requisitos [12]. O último requisito afirma que não existem transições (tarefas) mortas no estado inicial *i*.

A figura A.2 mostra uma WF-net que não é correcta. Existem várias deficiências. Se o *time_out_1* e o *processamento_2* ou o *time_out _2* e o *processamento_1* dispararem, a WF-net não irá terminar correctamente, porque um testemunho fica preso em *c4* ou *c5*. Se *time_out_1* e *time_out _2* dispararem, então a tarefa *processamento_NOK* será executada duas vezes e devido à presença de dois testemunhos em *o*, o momento da terminação não é claro.

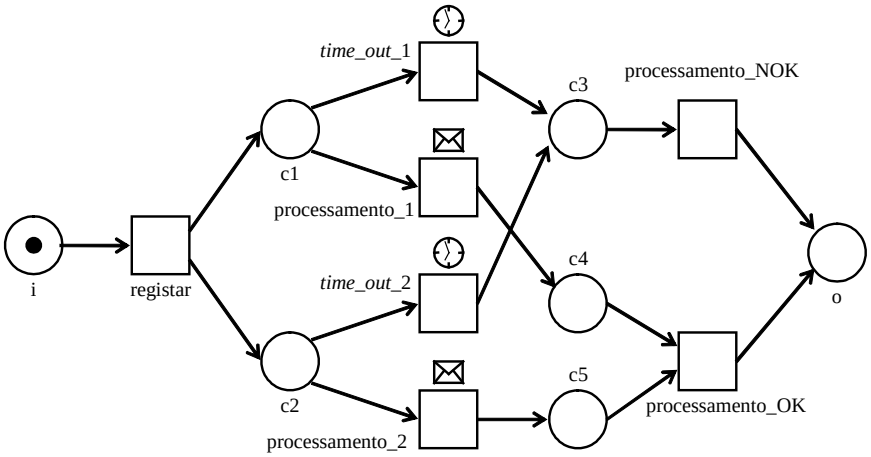


Figura A.2. Outra WF-net para o processamento de reclamações

Dada uma WF-net $PN = (P, T, F)$, queremos decidir se PN é correcta. Em [2] mostramos que a correcção corresponde à vivacidade e limitação. Para interligar a correcção à vivacidade e à limitação, definimos uma rede estendida $\underline{PN} = (\underline{P}, \underline{T}, \underline{F})$. $\underline{PN}$ é uma rede de Petri obtida através da adição de uma transição extra t^* que liga o e i. A rede de Petri estendida $\underline{PN} = (\underline{P}, \underline{T}, \underline{F})$ é definida como se segue: $\underline{P} = P$, $\underline{T} = T \cup \{t^*\}$, e $\underline{F} = F \cup \{<o, t^*>, <t^*, i>\}$. Mais a frente, chamaremos a tal rede estendida de rede *curto-circuito PN*. Uma rede curto-circuito permite a formulação do seguinte teorema.

TEOREMA 1. Uma WF-net PN é correcta sse $(\underline{PN}, i)$ é viva e limitada.

PROVA. Ver [2].

Este teorema mostra que as técnicas de análise baseadas em redes de Petri podem ser usadas para verificar a correcção.

A.4 Caracterização Estrutural da Correcção

O teorema 1 fornece uma caracterização útil do nível da qualidade da definição de um processo de *workflow*. No entanto, existem alguns problemas:

- Para uma WF-net complexa pode ser intratável decidir a sua correcção. (Para WF-nets arbitrárias, a vivacidade e a limitação são decidíveis mas ao mesmo tempo são também EXPSPACE–hard, cf. Cheng, Esparza, e Palsberg [7].);
- A correcção é um requisito mínimo. A capacidade de leitura ou de manutenção são tópicos que não são considerados pelo teorema 1; e
- O teorema 1 não mostra de que forma uma WF-net que não é correcta deve ser modificada, isto é, não identifica construções que invalidem a propriedade da correcção.

Estes problemas provêm do facto da definição de correcção estar relacionada com a dinâmica de uma WF-net, enquanto que o desenhador de workflows está preocupado com a estrutura estática de uma WF-net. Por isso, é interessante investigar caracterizações estruturais de WF-net correctas. Com este propósito, introduzimos três subclasses de WF-net interessantes: WF-net de escolha livre, WF-net bem estruturadas e WF-net com Cobertura-S.

A.4.1 WF-nets de escolha livre

A grande maioria dos WFMSs disponíveis de momento abstrai os estados entre as tarefas; por outras palavras, os estados não são representados explicitamente. Estes WFMSs usam blocos de construção tais como o AND-split, o AND-join, o OR-split, e o OR-join para especificar processos de *workflow*. O AND-split e o AND-join são usados para o encaminhamento paralelo. O OR-split e o OR-join são usados para o encaminhamento condicional. Porque estes sistemas abstraem-se dos estados, cada escolha é feita dentro de um bloco de construção OR-split. Se modelamos um OR-split em termos de uma rede de Petri, o OR-split corresponde a um número de transições que partilham o mesmo conjunto de lugares de entrada. Isto significa que para esses WFMSs, um processo de *workflow* corresponde a uma rede de Petri de escolha livre (cf. definição 7).

É fácil ver que a definição de um processo composta por AND-splits, AND-joins, OR-splits e OR-joins é de escolha livre. Se duas transições t_1 e t_2 partilham um lugar de entrada ($\bullet t_1 \cap \bullet t_2 \neq \emptyset$), então fazem parte de um OR-split, isto é, existe uma "escolha livre" entre um número de alternativas. Por isso, os conjuntos de lugares de entrada de t_1 e t_2 devem ser iguais ($\bullet t_1 = \bullet t_2$).

A figura A.2 mostra uma WF-net de escolha livre. A WF-net ilustrada na figura A.1 não é de escolha livre; *arquivar* e *processar_reclamação* partilham um lugar de entrada, mas os dois conjuntos de entradas correspondentes diferem.

Avaliámos vários WFMSs e apenas um desses sistemas (o COSA [18]) permite construções que são comparáveis a WF-nets de escolha não livre. Por isso, faz sentido considerar as redes de Petri de escolha livre em mais detalhe. Claramente, o paralelismo, o encaminhamento sequencial, o encaminhamento condicional e a iteração podem ser modelados sem violar a propriedade de escolha livre. Outra razão para restringir as WF-nets a redes de Petri de escolha livre é a seguinte: se permitimos a existência de redes de Petri de escolha não

livre, então a escolha entre tarefas conflituosas *pode ser* influenciada pela ordem em que as tarefas precedentes são executadas. O encaminhamento de um caso deve ser independente da ordem em que as tarefas são executadas. Uma situação em que a propriedade de escolha livre é violada consiste muitas vezes numa mistura de paralelismo e escolha. A figura A.3 mostra tal situação. O disparo da transição t_1 introduz paralelismo. Embora não exista uma escolha real entre t_2 e t_5 (t_5 não está pronta), a execução paralela de t_2 e t_3 resulta numa situação em que t_5 não tem permissão para ocorrer. Contudo, se a execução de t_2 for atrasada até t_3 ter sido executada, então existe uma escolha real entre t_2 e t_5. Na nossa opinião, o paralelismo deve ser separado da escolha entre duas ou mais alternativas. Por isso, consideramos a construção de escolha não livre ilustrada na figura A.3 imprópria. Na literatura, o termo *confusão* é muitas vezes usado para referir a situação ilustrada na figura A.3.

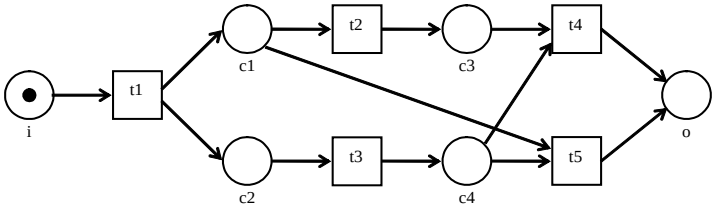


Figura A.3. Uma WF-net de escolha não livre contendo uma mistura de paralelismo e escolha

As redes de Petri de escolha livre têm sido estudadas extensivamente (cf. [9]) porque parecem ser um bom compromisso entre o poder expressivo e a capacidade de serem analisadas. É uma classe de redes de Petri para a qual existem fortes resultados teóricos e eficientes técnicas de análise. Por exemplo, o bem conhecido Teorema Rank ([8]) permite-nos formular o seguinte corolário.

COROLÁRIO 1. O seguinte problema pode ser resolvido em tempo polinomial: dada uma WF-net de escolha livre, decidir se esta está correcta.

PROVA. Seja *PN* uma WF-net de escolha livre. A rede curto-circuito $\underline{PN}$ é também de escolha livre. Por isso, o problema de decidir se ($\underline{PN}$, *i*) é viva e limitada pode ser resolvido em tempo polinomial (Teorema Rank [8]). Pelo teorema 1 isto corresponde a correcção.

O corolário 1 mostra que, para redes de escolha livre, existem algoritmos eficientes para decidir a correcção. Além disso, uma WF-net de escolha livre correcta é garantidamente segura (dado um estado inicial com apenas um testemunho em *i*).

LEMA 1. Uma WF-net de escolha livre correcta é segura.

PROVA. Seja *PN* uma WF-net de escolha livre correcta. $\underline{PN}$ é a rede de Petri *PN* estendida com uma transição que liga *o* e *i*. $\underline{PN}$ é de escolha livre e bem formada. Assim, $\underline{PN}$ tem Cobertura-S [9] (i.e., cada lugar é parte de um componente embebido e fortemente ligado de uma máquina de estados). Visto

que inicialmente existe apenas um testemunho, ($\underline{PN}$, i) é segura assim como (PN, i).

A segurança é uma propriedade recomendável porque não faz sentido ter múltiplos testemunhos num lugar que representam uma condição. Uma condição ou é verdadeira (um testemunho) ou falsa (sem testemunhos).

Embora a maioria dos WFMSs apenas permita workflows de escolha livre, as WF-nets de escolha livre não são uma caracterização estrutural satisfatória para identificar "bons" workflows. Por um lado, existem WF-nets de escolha não livre que correspondem a workflows adequados (cf. figura A.1), por outro lado, existem WF-nets de escolha livre e correctas que não fazem sentido. Contudo, a propriedade de escolha livre é uma propriedade recomendável. Se um *workflow* pode ser modelado como uma WF-net de escolha livre, então assim deverá ser feito. A especificação de um *workflow* baseada numa WF-net de escolha livre pode ser executada pela maioria dos sistemas de workflows. Além disso, uma WF-net de escolha livre permite técnicas de análise eficientes e é mais fácil de entender. Construções de escolhas não livres, tais como a construção mostrada na figura A.3, são potenciais fontes de comportamentos anómalos (ex., *deadlocks*) que são difíceis de identificar.

A.4.2 WF-nets bem formadas

Outra aproximação para obter uma caracterização estrutural de "bons" workflows, é equilibrar os AND/OR-splits e os AND/OR-joins. Claramente, dois fluxos paralelos iniciados por um AND-split não devem ser unidos com um OR-join. Dois fluxos alternativos criados através de um OR-split não devem ser sincronizados com um AND-join. Como mostra a figura A.4, um AND-split deve ser complementado por um AND-join e um OR-split deve ser complementado por um OR-join.

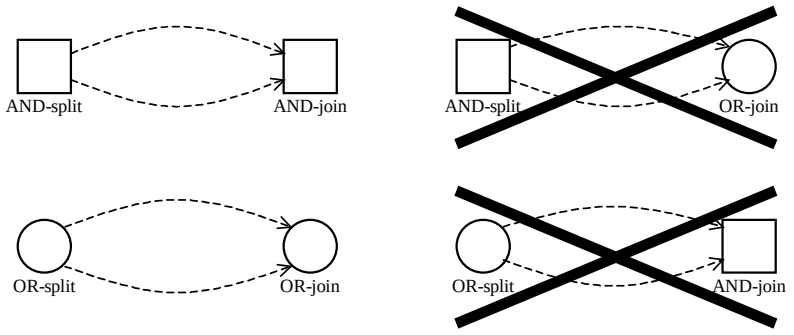


Figura A.4. Boas e más construções

Uma das deficiências da WF-net ilustrada na figura A.2 é o facto do AND-split *registar* ser complementado com um OR-join *c3* ou por um OR-join *o*. Para formalizar o conceito ilustrado na figura A.4 damos a seguinte definição.

DEFINIÇÃO 13 (Bem manuseada). Uma rede de Petri PN é bem manuseada sse, para qualquer par de nós x e y tais que um dos nós é um lugar e o outro uma

transição e para qualquer par de caminhos elementares C_1 e C_2 de x para y, $\alpha(C_1) \cap \alpha(C_2) = \{x, y\} \Rightarrow C_1 = C_2$.

Note-se que a WF-net ilustrada na figura A.2 não é bem manuseada. O bom manuseamento pode ser decidido em tempo polinomial aplicando uma versão modificada da técnica *max-flow min-cut*. Uma rede de Petri que é bem manuseada tem um número de propriedades interessantes como forte ligação e são bem formadas.

LEMA 2. Uma rede de Petri fortemente ligada e bem manuseada é bem formada.

PROVA. Seja *PN* uma rede de Petri bem manuseada e fortemente ligada. Claramente, não existem caminhos que tenham PT-handles nem TP-handles [11]. Por essa razão, a rede é estruturalmente limitada (ver teorema 3.1 em [11]) e estruturalmente viva (ver teorema 3.1 em [11]). Logo, *PN* é bem formada.

Claramente, o bom manuseamento é uma propriedade recomendável para qualquer WF-net *PN*. Além do mais, também necessitamos que a rede de curto-circuito $\underline{PN}$ seja bem manuseada. Impusemos este requisito adicional pela seguinte razão: supondo que queremos usar *PN* como parte de uma maior WF-net *PN'*.

PN' é a WF-net original estendida com uma "tarefa *undo*". Ver a figura A.5. A transição *undo* corresponde à tarefa *undo*, as transições t_1 e t_2 foram acrescentadas para tornar *PN'* uma WF-net. É indesejável que a transição *undo* viole a propriedade de bom manuseamento da rede original. Contudo, *PN'* é bem manuseada sse $\underline{PN}$ é bem manuseada. Então, necessitamos que $\underline{PN}$ seja bem manuseada. Usamos o termo *bem estruturada* para nos referirmos a WF-nets cuja extensão é bem manuseada.

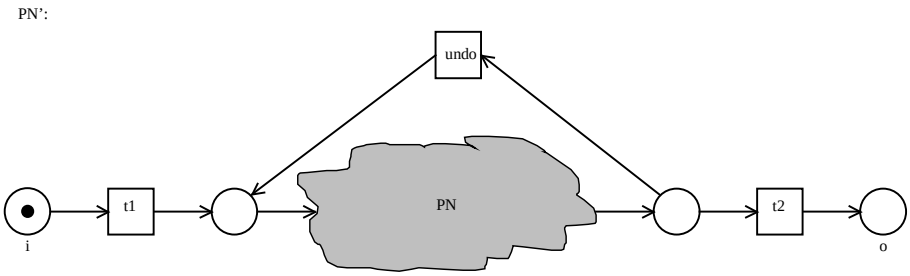


Figura A.5. A WF-net PN' é bem manuseada sse PN é bem manuseada

DEFINIÇÃO 14 (Bem estruturada). Uma WF-net *PN* é bem estruturada sse $\underline{PN}$ é bem manuseada.

WF-nets bem estruturadas têm um número de propriedades desejáveis. A correcção pode ser verificada em tempo polinomial e uma WF-net correcta e bem estruturada é segura. Para provar estas propriedades usamos alguns dos resultados obtidos para redes *non-self-controlling elementares e estendidas*.

DEFINIÇÃO 15 (Redes non-self-controlling elementares e estendidas). Uma rede de Petri *PN* é uma rede non-self-controlling elementar e estendida (ENSC) sse para todo o par de transições t_1 e t_2, tal que $\bullet t_1 \cap \bullet t_2 \neq \emptyset$, não existe um caminho elementar C ligando t_1 a t_2 tal que $\bullet t_1 \cap \alpha(C) = \emptyset$.

TEOREMA 2. Seja *PN* uma WF-net. Se *PN* é bem estruturada então $\underline{PN}$ é non-self-controlling elementar e estendida.

PROVA. Assumindo que $\underline{PN}$ não é non-self-controlling elementar e estendida. Isto significa que existe um par de transições t_1 e t_k tal que $\bullet t_1 \cap \bullet t_k \neq \emptyset$ e existe um caminho elementar $C = \langle t_1, p_2, t_2, \ldots, p_k, t_k \rangle$ de t_1 para t_k e $\bullet t_1 \cap \alpha(C) = \emptyset$. Seja $p_1 \in \bullet t_1 \cap \bullet t_k$. $C_1 = \langle p_1, t_k \rangle$ e $C_2 = \langle p_1, t_1, p_2, t_2, \ldots, p_k, t_k \rangle$ são caminhos de p_1 a t_k. (Note que C_2 é a concatenação de $\langle p_1 \rangle$ e C). Claramente, C_1 é elementar. Iremos também mostrar que C_2 é elementar. C é elementar, e $p_1 \notin \alpha(C)$ porque $p_1 \in \bullet t_1$. Logo, C_2 é também elementar. Uma vez que C_1 e C_2 são ambos caminhos elementares, $C_1 \neq C_2$ e $\alpha(C_1) \cap \alpha(C_2) = \{p_1, t_k\}$, concluímos que $\underline{PN}$ não é bem manuseada.

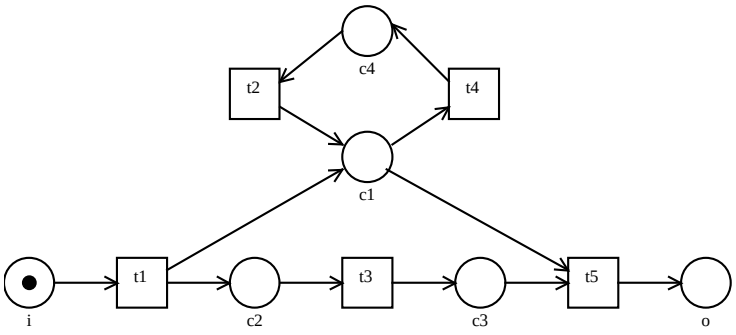


Figura A.6. Uma WF-net bem estruturada

Considere por exemplo a WF-net ilustrada na figura A.6. A rede WF é bem estruturada e, por isso, também é non-self-controlling elementar e estendida. No entanto, a rede não é de escolha livre. Contudo, é possível verificar a correcção de uma WF-net de uma forma muito eficiente.

COROLÁRIO 2. Dada uma WF-net bem estruturada, o problema de decidir se é correcta pode ser resolvido em tempo polinomial.

PROVA. Seja *PN* uma WF-net bem estruturada. A rede curto-circuito $\underline{PN}$ é non-self-controlling elementar e estendida (teorema 2) e estruturalmente limitada (ver prova do lema 2). Para non-self-controlling elementares e estendidas e limitadas, o problema de decidir se uma dada marcação é viva pode ser resolvido em tempo polinomial (ver [6]). Por isso, o problema de decidir se ($\underline{PN}$, i) é viva e limitada pode ser resolvido em tempo polinomial. Pelo teorema 1, isto corresponde a correcção.

LEMA 3. Uma WF-net correcta e bem estruturada é segura.

PROVA. Seja $\underline{PN}$ a rede *PN* estendida com uma transição que liga *o* e *i*. $\underline{PN}$ é non-self-controlling e estendida. $\underline{PN}$ é coberta por máquinas de estados (componente-S), ver corolário 5.3 em [6]. Logo, $\underline{PN}$ é segura tal como *PN* (ver prova de lema 1).

WF-nets bem estruturadas e WF-nets de escolha livre têm propriedades semelhantes. Em ambos os casos, a correcção pode ser verificada eficientemente e a correcção implica segurança. Apesar destas semelhanças, existem WF-nets bem estruturadas e correctas que não são de escolha livre (figura A.6.) e existem WF-nets correctas de escolha livre que não são bem estruturadas. De facto, é possível ter uma WF-net correcta que não é de escolha livre nem bem estruturada (figuras A.1 e A.3).

A.4.3 WF-nets com Cobertura-S

E sobre a correcção das WF-nets mostradas na figura A.1 e na figura A.3? A WF-net mostrada na figura A.3 pode ser transformada numa WF-net de escolha livre bem estruturada separando a escolha e o paralelismo. A WF-net ilustrada na figura A.1 não pode ser transformada numa WF-net de escolha livre ou WF-net bem estruturada sem produzir uma WF-net muito mais complexa. O lugar *c5* actua como uma espécie de acontecimento importante que é testado pela tarefa *processar_reclamação*. Os sistemas de gestão de *workflow* tradicionais que não descrevem o estado dos casos de forma explícita, não estão preparados para suportar o *workflow* especificado pela figura A.1. Apenas sistemas de gestão de *workflow* como o COSA [18] têm a capacidade de executar um *workflow* baseado em estados. Contudo, é interessante considerar generalizações de WF-nets de escolha livre e WF-net bem estruturadas: *WF-nets com Cobertura-S* podem ser vistas como uma generalização.

DEFINIÇÃO 16 (Cobertura-S). Uma WF-net tem Cobertura-S se a rede curto-circuito $\underline{PN}$ tem Cobertura-S.

As WF-nets ilustradas nas figuras A.1 e A.3 têm Cobertura-S. A WF-net ilustrada na figura A.2 não tem Cobertura-S. Os dois seguintes corolários mostram que ter Cobertura-S é uma generalização da propriedade escolha livre e bem estruturada.

COROLÁRIO 3. Uma WF-net de escolha livre correcta tem Cobertura-S.

PROVA. A rede curto-circuito $\underline{PN}$ é de escolha livre e bem formada. Logo, $\underline{PN}$ tem Cobertura-S (cf. [9]).

COROLÁRIO 4. Uma WF-net correcta e bem estruturada tem Cobertura-S.

PROVA. $\underline{PN}$ é non-self-controlling estendida (teorema 2). Assim, $\underline{PN}$ tem Cobertura-S (cf. corolário 5.3 em [6]).

Todas as WF-nets correctas existentes neste apêndice têm Cobertura-S. Toda a WF-net com Cobertura-S é segura. A única WF-net que não é correcta é a WF-net ilustrada na figura A.2, que não tem Cobertura-S. Estes e outros exemplos indicam que existe uma elevada correlação entre a Cobertura-S e a correcção. Parece que a Cobertura-S é um dos requisitos básicos que qualquer definição de processo de *workflow* deve satisfazer. De um ponto de vista formal, é possível construir WF-nets que são correctas mas que não têm Cobertura-S. Tipicamente, essas redes contêm lugares que não restringem o disparo de uma transição, mas que não estão em nenhum componente-S. Veja, por exemplo, a figura 65 em [14]. De um ponto de vista prático, essas WF-nets devem ser evitadas. As WF-nets que não possuem Cobertura-S são difíceis de interpretar porque as propriedades estruturais e dinâmicas não coincidem. Por exemplo, essas redes podem ser vivas e limitadas mas não estruturalmente limitadas. Parece não haver necessidade prática para usar construções que violam a propriedade da Cobertura-S. Por isso, consideramos a Cobertura-S um requisito básico que qualquer WF-net deve satisfazer.

Outra forma de olhar para a Cobertura-S tem a seguinte interpretação: Componentes-S correspondem ao *fluxo de documentos*. Para suportar um *workflow*, diversos fragmentos de informação são criados, usados e actualizados. Podemos pensar nesses fragmentos de informação como documentos físicos de tal forma que em qualquer instante o documento está num lugar na WF-net. Naturalmente, a informação num documento pode ser copiada para outro documento enquanto executa uma tarefa (i.e. transição) processando ambos os documentos. Inicialmente, todos os documentos estão presentes mas um documento pode estar vazio (i.e., corresponde a uma folha de papel em branco). É fácil de ver que o fluxo de um destes documentos corresponde a uma máquina de estados (assumindo a existência de uma transição t^*). Esses fluxos de documentos sincronizam-se através de tarefas do tipo *join*. Desta forma a composição destes fluxos produz uma WF-net com Cobertura-S. Podemos pensar nos fluxos de documentos como *threads*. Considere por exemplo a rede curto-circuito da WF-net ilustrada na Figura A.1. Esta rede pode ser composta pelos dois seguintes *threads*: (1) um *thread* corresponde ao processamento do questionário (lugares *i*, *c*2, *c*3, *c*5, e *o*), e (2), um *thread* corresponde ao actual processamento da reclamação (lugares *i*, *c*2, *c*4, *c*5, *c*6, *c*7, *c*8 e *c*9). Note-se que as tarefas *registar* e *arquivar* são usadas em ambos os *threads*.

Embora uma WF-net possa, em princípio, ter exponencialmente muitos Componentes-S, eles são muitos fáceis de computar para workflows encontrados na prática (veja também a interpretação anterior de Componentes-S como fluxos de documentos ou *threads*). Note-se que a Cobertura-S apenas depende da estrutura e o grau de ligação é geralmente baixo (i.e., a matriz de incidência de uma WF-net tem tipicamente poucas entradas vazias). Infelizmente, em geral não é possível verificar a correcção de uma WF-net com Cobertura-S em tempo polinomial. O problema de decidir a correcção de uma WF-net com Cobertura-S é PSPACE-*complete*. Para a maioria das aplicações este não é um problema real. Na maioria dos casos, o número de tarefas numa definição de processo de *workflow* é menor que 100 e o número de estados é menor que 200.000. As ferramentas que usam técnicas *standards*, tais como a construção de um grafo de cobertura, não têm nenhum problema em lidar com estas definições de processo de *workflow*.

A.4.4 Sumário

As três caracterizações estruturais (escolha livre, bem estruturada e Cobertura-S) mostraram ser muito úteis para a análise da definição de processos de *workflow*. Com base na nossa experiência, temos boas razões para acreditar que a Cobertura-S é uma propriedade desejável que qualquer definição de *workflow* deve satisfazer. As construções que violem a Cobertura-S podem ser detectadas facilmente e podem ser construídas ferramentas para ajudar o designer a construir uma WF-net com Cobertura-S. A Cobertura-S é uma generalização da boa estruturação e a propriedade de escolha livre (corolários 3 e 4). Ambas, a boa estruturação e a propriedade de escolha livre, correspondem a propriedades desejáveis de um *workflow*. Uma WF-net que satisfaz pelo menos uma destas duas propriedades pode ser analisada muito eficientemente. Contudo, mostramos que existem workflows que não são de escolha livre, nem bem estruturados. Considere por exemplo a figura A.1. O facto da tarefa *processa_reclamação* testar se existe um testemunho em *c*5, impende a WF-net de ser de escolha livre ou bem estruturada. Embora este seja um *workflow* bastante adequado, a maioria dos sistemas de gestão de *workflow* não suportam uma construção de encaminhamento tão avançado. Mesmo se formos capazes de utilizar workflows baseados em estados (ex., o COSA) permitindo construções que violam uma boa estruturação e a propriedade de escolha livre, então as caracterizações estruturais continuam a ser úteis. Se uma WF-net não é de escolha livre nem bem estruturada, devemos localizar a fonte que viola uma dessas propriedades e verificar se é realmente necessário usar uma rede de escolha não livre ou uma construção que não é bem formada. Se uma escolha não livre ou uma construção não bem estruturada é realmente necessária, então a correcção da construção deve ser duplamente verificada, porque é uma potencial fonte de erros. Desta forma, a legibilidade e a manutenção de uma definição de um processo de *workflow* podem ser melhoradas.

A.5 Composição de WF-nets

As WF-nets mostradas neste apêndice são muito simples comparadas com os workflows encontrados na prática. Por exemplo, num cenário prático existem workflows que consistem em mais de 100 tarefas com uma estrutura de interacção muito complexa. Para o designer destes workflows a complexidade é esmagadora e a comunicação com os utilizadores finais usando enormes diagramas é difícil. Na maioria dos casos, a decomposição hierárquica é utilizada para ultrapassar este problema. Um *workflow* complexo é decomposto em sub fluxos e cada sub fluxo é descomposto em pequenos sub fluxos até o nível desejado de detalhe ser alcançado. Muitos WFMSs permitem tal decomposição hierárquica. Além disso, este mecanismo pode ser utilizado para a reutilização de workflows existentes. Considere, por exemplo, múltiplos workflows que partilham um subprocesso genérico. Alguns vendedores de WFMSs também fornecem modelos de referência que correspondem a workflows típicos existentes em seguradoras, no sector financeiro, no marketing, em compras, em procuração, em logística e na manufacturação.

Os modelos de referência, a reutilização e a estruturação de workflows complexos requerem o conceito de hierarquia. O conceito mais comum de hierarquia apoiado por muitos WFMSs é o *refinamento de tarefas* (uma tarefa

pode ser refinada num sub fluxo). Este conceito é ilustrado na figura A.7. A WF-net PN_1 contem uma tarefa t^+ que é refinada por outra WF-net PN_2; por outras palavras, t^+ já não é uma tarefa mas sim uma referência para um sub fluxo. Uma WF-net que representa um sub fluxo deve satisfazer os requisitos como de uma WF-net normal. A semântica do conceito de hierarquia é clara; substitui simplesmente a transição refinada pela sub rede correspondente. A figura A.7 mostra que o refinamento de t^+ em PN_1 por PN_2 produz uma WF-net PN_3.

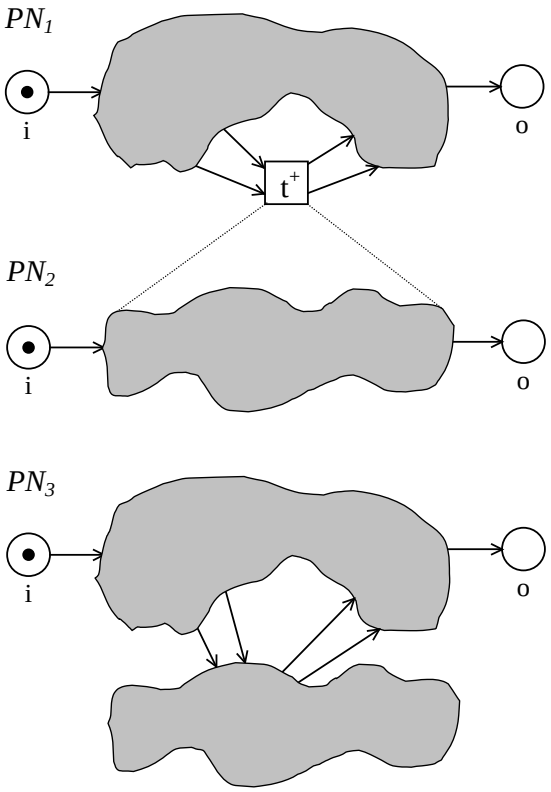


Figura A.7. Refinamento de tarefas: A WF-net PN_3 é composta de PN_1 e PN_2

O conceito de hierarquia pode ser explorado para estabelecer a correcção de um *workflow*. Dado um modelo hierárquico complexo de um *workflow*, é possível verificar a sua correcção pela análise separada de cada um dos seus sub fluxos. Isto é ilustrado pelo seguinte teorema.

TEOREMA 3. Seja $PN_1 = (P_1, T_1, F_1)$ e $PN_2 = (P_2, T_2, F_2)$ duas WF-nets tais que $T_1 \cap T_2 = \emptyset$, $P_1 \cap P_2 = \{i, o\}$, e $t+ \in F_1$. Seja $PN_3 = (P_3, T_3, F_3)$ a WF-net obtida pela substituição da transições $t+$ em PN_1 por PN_2, i.e., $P_3 = P_1 \cup P_2$, $T_3 = (T_1 \setminus \{t+\}) \cup T_2$, e $F_3 = \{(x, y) \in F_1 \mid x \neq t+ \wedge y \neq t+\} \cup \{(x, y) \in F_2 \mid \{x, y\} \cap \{i, o\} = \emptyset\} \cup \{(x,$

$y) \in P_1 \times T_2 \mid (x, t+) \in F_1 \wedge (i, y) \in F_2\} \cup \{(x, y) \in T_2 \times P_1 \mid (t+, y) \in F_1 \wedge (x, o) \in F_2\}$. ($PN_1$, i) e (PN_2, i) são seguras e correctas sse (PN_3, i) é segura e correcta.

PROVA. A prova é um caso especial da prova do teorema 3 em [5]. O ponto crucial da prova é a observação que todos os estados em PN_3 podem ser atribuídos a um estado em PN_1 e um estado em PN_2 e vice-versa. Além disso, é essencial que as redes sejam seguras: se a sub rede PN_2 é activada múltiplas vezes, o seu comportamento não pode ser relacionado a um único disparo de $t+$ em PN_1. Para mais detalhes consultar [5].

O teorema 3 é uma generalização do resultado dado por Vallette em [16]. A figura A.8 mostra um processo de *workflow* hierárquico com um fluxo de trabalho principal e dois sub-processos. Ambos os sub-processos são seguros e correctos. Se no *workflow* principal os dois sub-processos forem substituídos por tarefas normais, então o *workflow* principal será também seguro e correcto. Por isso, o *workflow* total mostrado na figura A.8 será também seguro e correcto. O teorema 3 é de particular importância para a reutilização de sub-processos. Para a análise de um *workflow* complexo, todos os sub fluxos seguros e correctos podem ser considerados como uma tarefa única. Isto permite uma análise modular eficiente da propriedade de correcção.

Os resultados apresentados neste apêndice fornecem aos designers de *workflow* uma ajuda para construir workflows correctos. Embora seja possível usar instrumentos padrão de análise baseadas em redes de Petri, desenvolvemos um analisador de *workflow*, chamado Woflan, que pode ser usado por pessoas que não estejam familiarizadas com a teoria das redes de Petri [4, 17]. O Woflan possui interfaces com produtos de *workflow* existentes tal como o Staffware, o COSA, o METEOR e o Protos.

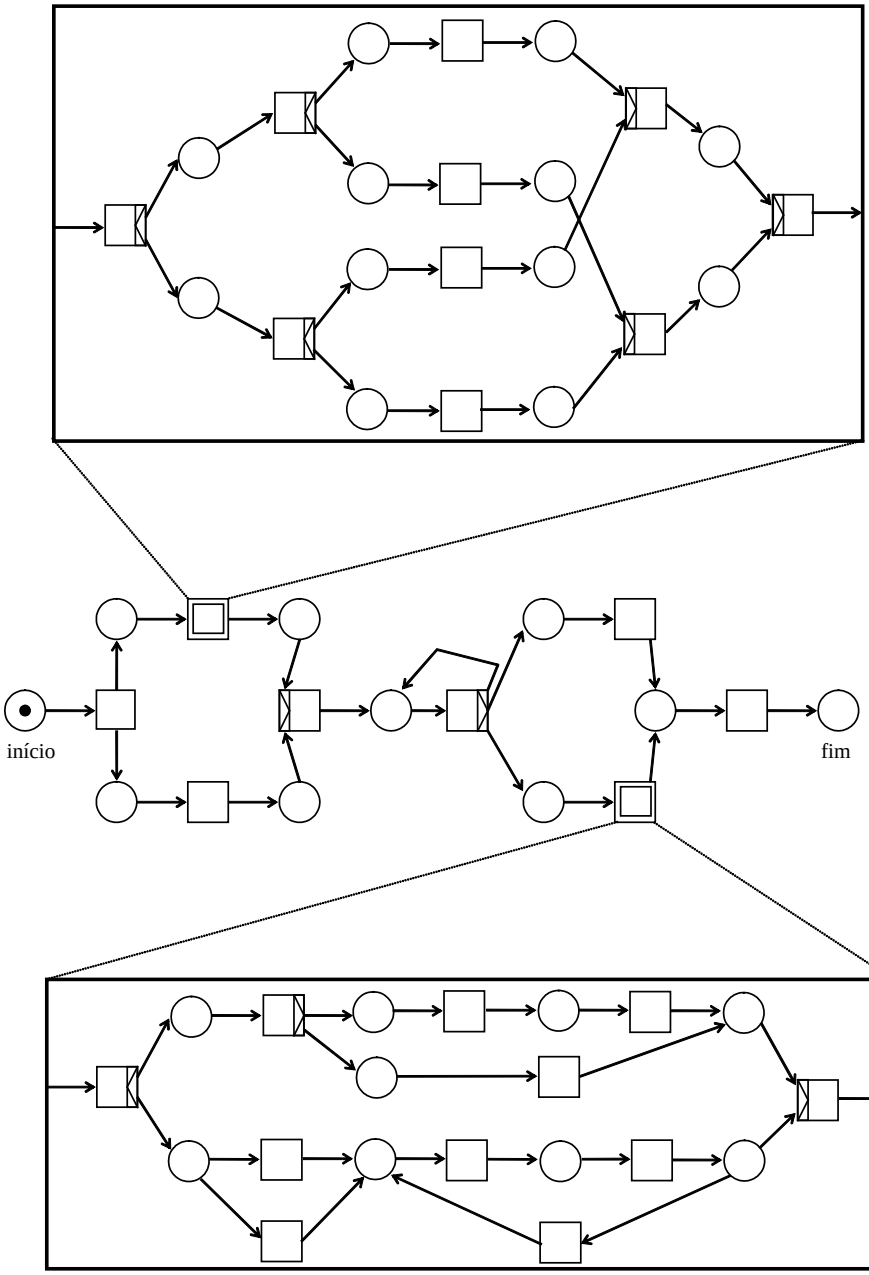


Figura A.8. Construir workflows complexos (que são seguros e correctos) a partir de sub-processos seguros e correctos

A.6 Referências

[1] W.M.P. van der Aalst. Three Good Reasons for Using a Petri-net-based Workflow Management System. In S. Navathe and T. Wakayama, editors, *Proceedings of the International Working Conference on Information and Process Integration in Enterprises (IPIC'96)*, pages 179-201, Cambridge, Massachusetts, Nov 1996.

[2] W.M.P. van der Aalst. Verification of Workflow Nets. In P. Azéma and G. Balbo, editors, *Application and Theory of Petri Nets 1997*, volume 1248 of *Lecture Notes in Computer Science*, pages 407-426. Springer-Verlag, Berlin, 1997.

[3] W.M.P. van der Aalst. The Application of Petri Nets to Workflow Management.
The Journal of Circuits, Systems and Computers, 8(1):21-66, 1998.

[4] W.M.P. van der Aalst. Woflan: A Petri-net-based Workflow Analyzer.
Systems Analysis - Modelling - Simulation, 35(3):345-357, 1999.

[5] W.M.P. van der Aalst. Finding Control-Flow Errors Using Petri-Net-Based Techniques. *Business Process Management: Models, Techniques, and Empirical Studies*, volume 1806 of *Lecture Notes in Computer Science*, pages 161-183. Springer-Verlag, Berlin, 2000.

[6] K. Barkaoui, J.M. Couvreur, and C. Dutheillet. On liveness in Extended Non Self-Controlling Nets. In G. De Michelis and M. Diaz, editors, *Application and Theory of Petri Nets 1995*, volume 935 of *Lecture Notes in Computer Science*, pages 25-44. Springer-Verlag, Berlin, 1995.

[7] A. Cheng, J. Esparza, and J. Palsberg. Complexity results for 1-safe nets. In R.K. Shyamasundar, editor, *Foundations of software technology and theoretical computer science*, volume 761 of *Lecture Notes in Computer Science*, pages 326-337. Springer-Verlag, Berlin, 1993.

[8] J. Desel. A proof of the Rank theorem for extended free-choice nets. In K. Jensen, editor, *Application and Theory of Petri Nets 1992*, volume 616 of *Lecture Notes in Computer Science*, pages 134-153. Springer-Verlag, Berlin, 1992.

[9] J. Desel and J. Esparza. *Free Choice Petri Nets*, volume 40 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge, UK, 1995.

[10] J. Esparza. Synthesis rules for Petri nets, and how they can lead to new results.
In J.C.M. Baeten and J.W. Klop, editors, *Proceedings of CONCUR 1990*, volume 458 of *Lecture Notes in Computer Science*, pages 182-198. Springer-Verlag, Berlin, 1990.

[11] J. Esparza and M. Silva. Circuits, Handles, Bridges and Nets. In G. Rozenberg, editor, *Advances in Petri Nets 1990*, volume 483 of *Lecture Notes in Computer Science*, pages 210-242. Springer-Verlag, Berlin, 1990.

[12] K. Gostellow, V. Cerf, G. Estrin, and S. Volansky. Proper Termination of Flow-of-control in Programs Involving Concurrent Processes. *ACM Sigplan*, 7(11):15-27, 1972.

[13] E. Kindler and W.M.P. van der Aalst. Liveness, Fairness, and Recurrence. *Information Processing Letters*, 70: 269-274, 1999.

[14] W. Reisig. *Petri Nets: An Introduction*, volume 4 of *EATCS Monographs in Theoretical Computer Science*. Springer-Verlag, Berlin, 1985.

[15] W. Reisig and G. Rozenberg, editors. *Lectures on Petri Nets I: Basic Models*, volume 1491 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1998.
[16] R. Vallete. Analysis of Petri Nets by Stepwise Refinements. *Journal of Computer and System Sciences*, 18:35-46, 1979.
[17] H.M.W. Verbeek, T. Basten, and W.M.P. van der Aalst. Diagnosing Workflow Processes using Woflan. Computing Science Report 99/02, Eindhoven University of Technology, Eindhoven, 1999.
[18] Software-Ley. *COSA User Manual*. Software-Ley GmbH, Pullheim, Germany, 1998.

(Página deixada propositadamente em branco)

APÊNDICE B.

MODELAR WORKFLOWS COM O UML

Recentemente, o *Unified Modeling Language* (UML) tornou-se a ferramenta padrão para o desenvolvimento de *software*. O UML é uma linguagem gráfica para visualizar, especificar, construir e documentar os artefactos de um sistema de *software*. No entanto, a utilização do UML não é restrita apenas ao desenvolvimento de *software*. Alguns dos seus diagramas são utilizados para a modelação de organizações, engenharia de negócio, análise de processos e configuração de sistemas. Dada a grande utilização do UML como um padrão da indústria e pelo facto do UML oferecer quatro tipos de diagramas para a modelação de processos, este apêndice discute o uso do UML no contexto da gestão de workflows. São apresentados os tipos de diagramas mais relevantes e é discutida a relação com a técnica de modelação utilizada neste livro.

O desenvolvimento do UML começou em 1994 quando James Rumbough juntou-se a Grady Booch da Rational Software Corporation. Ambos tinham vindo a trabalhar em métodos orientados a objectos chamados de OMT (*Object Modeling Technique*) e Booch. Em 1994 existiam cerca de cinquenta métodos orientados a objectos. Rumbough e Booch juntaram forças para unificar os seus métodos e para ganhar massa crítica. Em 1995, um terceiro autor promissor de métodos orientados a objectos, juntou-se a esta iniciativa: Ivar Jacobson contribuiu com o seu trabalho em OOSE (*Object-Oriented Software Engineering*) para o projecto UML da Rational. Em Janeiro de 1997, o UML 1.0 foi oferecido ao *Object Management Group* (OMG) em resposta ao pedido de uma linguagem de modelação standard. Desde esta altura, o UML foi adoptado pela indústria e pelo meio académico como a linguagem *standard* para a modelação orientada a objectos. Adicionalmente, a linguagem foi estendida e refinada em diversas iterações. Este apêndice é baseado no UML 1.3.

O UML 1.3 define os seguintes tipos de diagramas:

- Diagrama de casos de utilização
- Diagrama de classes
- Diagrama de sequência
- Diagrama de colaboração
- Diagrama de estados
- Diagrama de actividades
- Diagrama de componentes
- Diagrama de instalação

Um *diagrama de casos* de utilização mostra um conjunto de casos e actores e as relações entre estes. Um *diagrama de classes* mostra um conjunto de classes e as relações entre estas. Ambos os diagramas abordam a visão estática do sistema. O diagrama de casos de utilização centra-se em partes de funcionalidades identificáveis e coloca-as em contexto. O *modelo de classes* é essencialmente um mecanismo para estruturar objectos. Tanto os *diagramas de sequência* como os *diagramas de colaboração* são essencialmente diagramas de interacção, isto é, diagramas centrados na interacção (ex., passagem de mensagens) entre objectos e actores. Um diagrama de sequência é um diagrama de interacção que se centra na ordenação temporal das mensagens. Um diagrama de colaboração centra-se mais na organização estrutural do que na ordenação temporal. Os *diagramas de estados* são tipicamente utilizados para modelar o ciclo de vida dos objectos. Um diagrama de estados centra-se nos estados dos objectos. Os *diagramas de actividades* são tipicamente utilizados para descrever o fluxo de controlo entre objectos. Comparado com os diagramas de estados o foco é passado dos estados para as actividades. Note que o UML utiliza quatro tipos de diagramas para modelar a perspectiva dinâmica de um sistema: diagramas de sequência, diagramas de colaboração, diagramas de estados e diagramas de actividades modelam o comportamento dinâmico. Os dois tipos de diagramas restantes modelam a perspectiva de implementação do sistema. Num *diagrama de componentes*, são agrupados conjuntos de objectos em componentes. Um *diagrama de instalação* mostra a configuração dos nós que realizam processamento e descreve os componentes que residem nesses nós.

Os sistemas de gestão de *workflow* centram-se na perspectiva do processo. Pelo facto dos diagramas de sequência, diagramas de colaboração, diagramas de estado e diagramas de actividade modelarem o comportamento dinâmico de um sistema, estes diagramas são muitos relevantes para a gestão de workflows e vão ser discutidos em maior detalhe neste capítulo. Os diagramas de componentes e os diagramas de instalação são relevantes para a arquitectura, a implementação e a configuração dos sistemas de *workflow*. Embora seja relevante, uma discussão detalhada destes tipos de diagramas está fora do âmbito deste livro. Os diagramas de casos de utilização são muito úteis nas primeiras etapas da modelação de workflows. Um diagrama de casos de utilização pode ser usado para identificar os *stakeholders* e clarificar os tipos de casos tratados pelo sistema de *workflow*. O diagrama de classes pode ser usado para modelar a relação entre os casos e os seus atributos.

B.1 Diagrama de Sequência

A figura B.1 mostra dois diagramas de sequência. O diagrama do lado esquerdo modela um cenário que descreve a encomenda bem sucedida de um livro por um cliente. O diagrama do lado direito modela um cenário em que a encomenda do cliente é rejeitada por não existir o livro encomendado em *stock*. Um diagrama de sequência mostra para cada objecto ou actor uma linha de vida. Em ambos os diagramas da figura B.1 existem três linhas de vida: a linha de vida do cliente, a linha de vida da livraria e a linha de vida da editora. O tempo aumenta ao longo de cada linha de vida de cima para baixo. Um diagrama de sequência também mostra as mensagens trocadas. Considere, por exemplo, o diagrama do lado esquerdo. Primeiro, o cliente encomenda um livro

enviando a mensagem *Encomenda_livro*. Depois a livraria (*on-line*) envia um pedido à editora para ver se o livro está disponível (mensagem *Consulta*). A editora responde enviando a mensagem *Em_stock* indicando que o livro está disponível. A livraria confirma a encomenda (mensagem *Confirma_encomenda*) e paga o livro (mensagem *Pagamento*). Após receber o pagamento, a editora envia o livro para o cliente (mensagem *Entrega_livro*) e notifica a livraria (mensagem *Notifica*). Com base nesta notificação (disparo), a livraria envia a factura (mensagem *Factura*) e o cliente paga o livro (mensagem *Pagamento*).

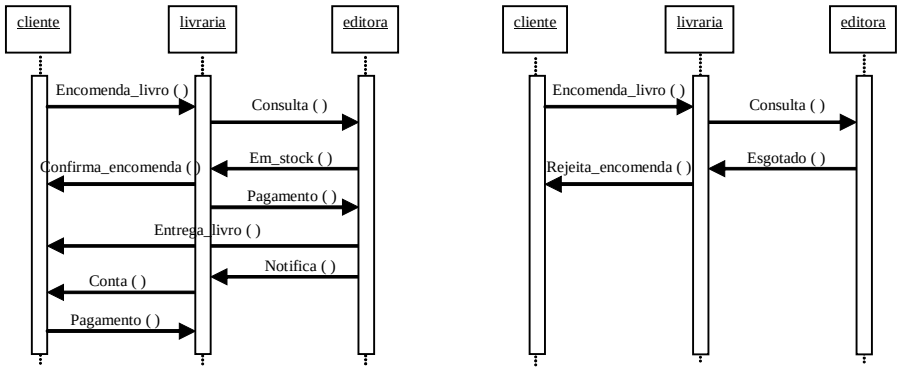


Figura B.1. Dois diagramas de sequência

Note que o diagrama do lado esquerdo não especifica um processo mas meramente um cenário. Este cenário corresponde ao tratamento de uma encomenda efectuada por um cliente com sucesso. Se o livro não estiver em *stock*, aplica-se o diagrama da direita. No segundo cenário, o livro não está disponível (mensagem *Esgotado*) e a encomenda do cliente é rejeitada (mensagem *Rejeita_encomenda*). A figura B.1 ilustra que os diagramas de sequência apenas podem ser usados para modelar cenários e não são adequados para fazer modelos de processos completos.

O diagrama de sequência básico não está preparado para o desenho de encaminhamentos tais como escolhas, sincronização, iteração, etc. Os diagramas de sequência foram estendidos com funcionalidades para representarem estas construções de encaminhamentos. No entanto, estes diagramas estendidos tornaram-se difíceis de ler e difíceis de interpretar.

B.2 Diagrama de Colaboração

Um diagrama de colaboração centra-se na organização de objectos que participam numa interacção. Comparado com os diagramas de sequência, a ênfase transita das relações temporais para as relações organizacionais. Do ponto de vista semântico, os diagramas de colaboração e os diagramas de sequência são permutáveis, ou seja, semanticamente equivalentes. As linhas de vida são substituídas por sequências numeradas. Considere a figura B.2. Os dois diagramas de colaboração correspondem aos dois diagramas de sequência

mostrados na figura B.1. É possível traduzir um diagrama de sequência num diagrama de colaboração sem perder qualquer informação (e vice-versa).

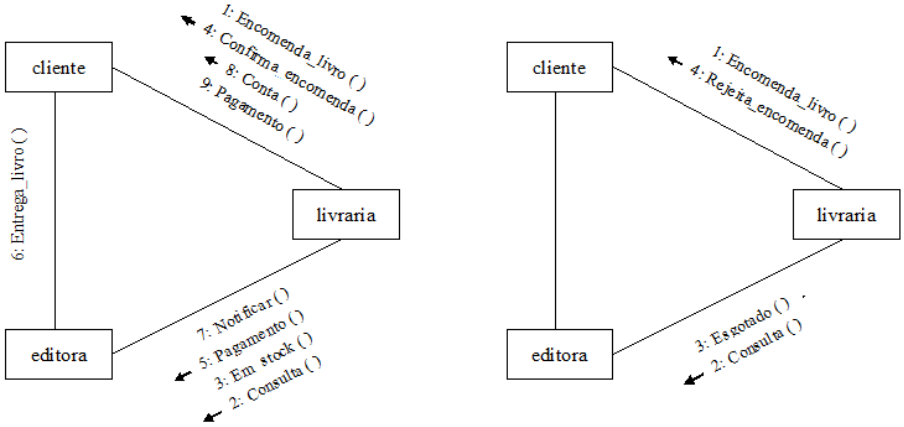


Figura B.2. Dois diagramas de colaboração

A ordem das mensagens trocadas é capturada por um esquema de numeração. Os números mostrados na figura B.2 indicam a ordem em que as mensagens são trocadas entre o cliente, a livraria e a editora. Os diagramas de colaboração podem ser estendidos com construções mais complexas tais como aninhamento, iteração e ramificação. No entanto, tal como os diagramas de sequência, os diagramas de colaboração são particularmente adequados para a modelação de cenários, isto é, exemplos de fluxos de controlo ordenados e sequenciados. Para modelar processos devem-se utilizar diagramas de estados ou diagramas de actividades.

B.3 Diagrama de Estados

Os diagramas de estados são extensões de máquinas de estados básicas. Uma máquina de estados básica é composta por estados e transições. Em qualquer momento, o sistema (ou objecto) reside num desses estados. Uma transição desloca um sistema de um estado para outro. A máquina de estados básica corresponde à classe de redes de Petri onde cada transição tem um lugar de entrada e um lugar de saída. Num diagrama de estados, este pode conter estados compostos, regiões ortogonais, variáveis, eventos, condições e acções. Estados compostos podem ser utilizados para representar aninhamento. As regiões ortogonais podem ser utilizadas para modelar paralelismo. As transições podem ser estendidas com regras chamadas ECA (Evento-Condição-Acção). Isto significa que uma transição apenas acontece quando ocorre um evento específico e uma condição é satisfeita. Tanto o evento como a condição são opcionais. É também possível adicionar uma acção a uma transição. Isto significa que a acção é executada no momento em que ocorre a transição. A notação padrão para estas regras ECA é "*evento[condição]/acção*."

A figura B.3 mostra um diagrama de estados muito simples. Este diagrama de estados modela o ciclo de vida de uma encomenda. O estado inicial é modelado por um ponto preto. O estado final é modelado por um ponto preto dentro de um círculo. Um estado é modelado por um rectângulo arredondado e as transições são modeladas por arcos. A transição que liga os estados *encomenda_criada* e *consulta_enviada* gera a acção *envia_consulta*. No estado *consulta_enviada*, duas potenciais transições são activadas, uma delas é iniciada pelo evento *notifica_em_stock* e leva ao estado *em_stock*, a outra é iniciada pelo evento *notifica_esgotado* e leva ao estado *esgotado*.

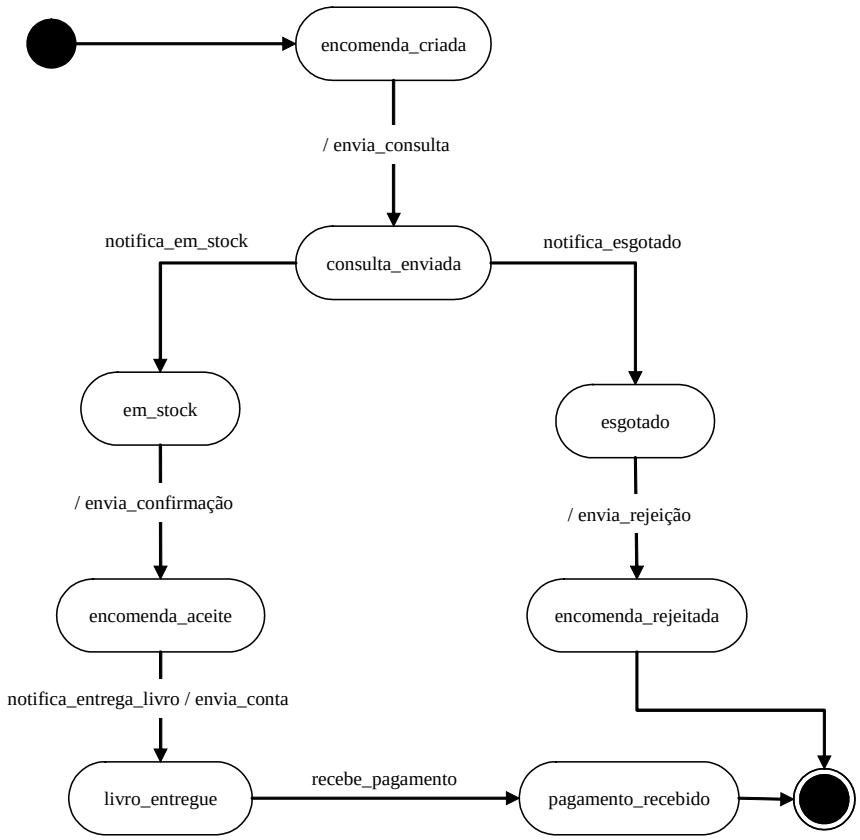


Figura B.3. Um diagrama de estados

B.4 Diagrama de Actividades

Os diagramas de estados são adequados para modelar o ciclo de vida de um objecto, mas, infelizmente, são menos adequados para modelar o controlo de fluxo entre objectos. Para este propósito, o UML oferece os diagramas de actividades. Estes diagramas são semelhantes às técnicas utilizadas neste livro, por isso, não é surpresa nenhuma ver diagramas de actividades a serem

utilizados em modelação de organizações, modelação de workflows e reengenharia de processos de negócio.

Considere a figura B.4. Este diagrama de actividades modela o processo ilustrado por dois diagramas de sequência/colaboração. O diagrama é dividido em três partes fundamentais: cliente, livraria e editora. Estas partes são chamadas *swimlanes*. Uma *swimlane* especifica um grupo de actividades e é particularmente útil na modelação de negócios. Utilizando as *swimlanes* é possível dividir um processo em papéis ou unidades organizacionais. Note-se que a técnica de modelação utilizada neste livro pode também ser estendida com *swimlanes*.

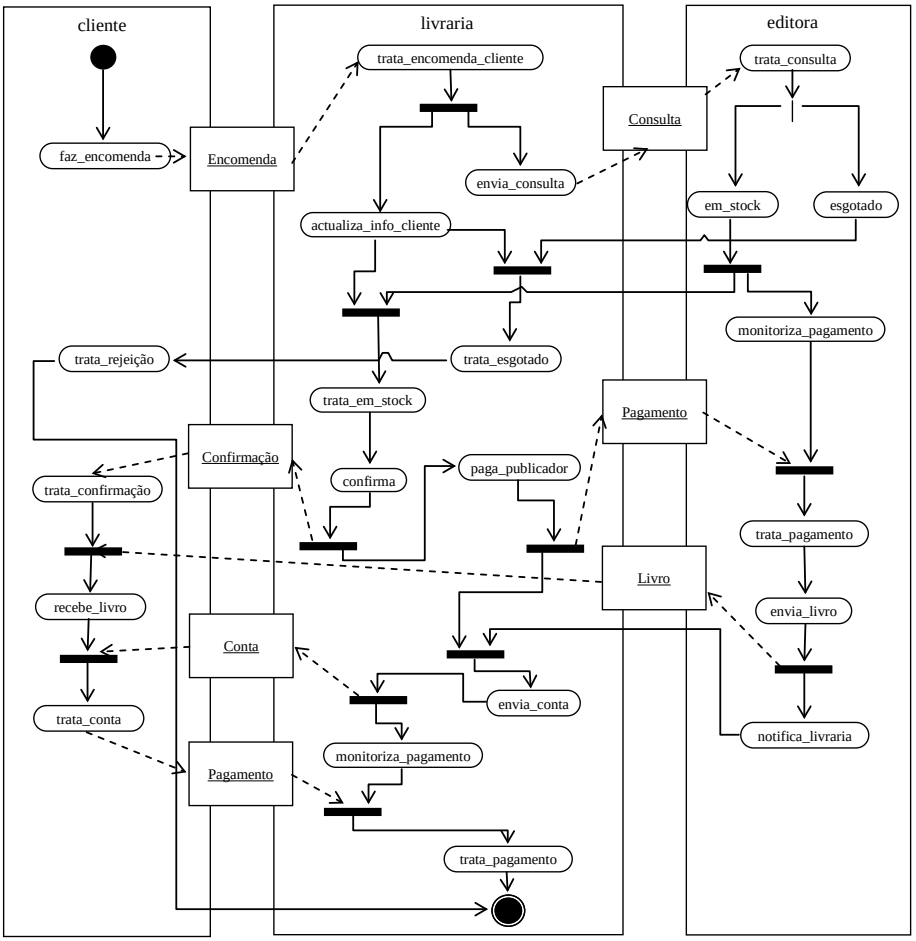


Figura B.4. Um diagrama de actividades

Tal como os diagramas de estados, os estados inicial e final são indicados com pontos pretos. As actividades (também chamadas de estados de actividade) são denotadas por rectângulos arredondados. As linhas sólidas correspondem ao controlo de fluxo e as linhas tracejadas correspondem ao fluxo dos objectos. Os

objectos transferidos são modelados por rectângulos. Considere, por exemplo, o canto superior esquerdo do diagrama de actividades da figura B.4. Começando no estado inicial, a actividade *faz_encomenda* é executada. Após a execução da actividade *envia_encomenda*, um objecto encomenda é transferido para a livraria que executa *trata_encomenda_cliente*. As linhas horizontais mais espessas na figura B.4 correspondem a *barras de sincronização*. Uma barra de sincronização pode ser um *fork* ou um *join*. Os *forks* correspondem a AND-splits. Os *joins* correspondem a AND-joins. Um OR-split explícito é modelado por um *ramo* e ilustrado por um losango. O losango também pode ser utilizado para modelar OR-joins. O diagrama de actividades da figura B.4 tem um ramo. Este ramo torna o processo dependente da disponibilidade do livro encomendado pelo cliente. As restantes partes do processo são fáceis de interpretar.

B.5 Outras Técnicas de Modelação de Processos

Muitas técnicas de modelação de processos têm sido desenvolvidas desde o princípio dos anos sessenta. Algumas destas técnicas são informais no sentido em que os diagramas utilizados não têm uma semântica formalmente definida. Estes modelos são tipicamente muito intuitivos e a interpretação varia consoante o designer, o domínio da aplicação e as características dos processos de negócio. Exemplos de técnicas informais incluem o ISAC, o DFD, o SADT e o IDEF. O SADT, e a sua equivalente militar IDEF0, foram desenvolvidas para descrever sistemas complexos e controlar o desenvolvimento de *software* complexo através de uma abordagem sistemática da definição dos requisitos. Um dos objectivos era desenvolver um processo que inclui a definição de papéis humanos e dos procedimentos interpessoais como parte da técnica. O SADT (ou IDEF) aborda a definição de requisitos através de uma série de passos que determinam *porquê* é que o sistema é necessário, *para que* irão servir as funcionalidades do sistema e *como* será construído o sistema. Técnicas relacionadas e comparáveis incluem o desenho estruturado de Yourdon, Structured Analysis de De Marco, Essential System Analysis de McMenamin e Palmer, e Information Systems Work and Analysis of Change (ISAC) desenvolvidas por Lundeberg, Goldkuhl e Nilson. Estas técnicas têm em comum o facto de não terem uma semântica formal. Apesar dos esforços feitos para fornecer uma semântica formal a estas técnicas (o mais notável foi o IDEF0), estas semânticas usam tipicamente uma interpretação que é diferente da forma como estes modelos são descritos em livros e aplicados na prática.

Existem várias técnicas formais de modelação de processos, por exemplo, as máquinas de estados finitos, sistemas de transição legendados, os diagramas de estados, as redes de Petri e as álgebras de processos tais como o ACP, o CSP e o CCS. As máquinas de estados finitos e os sistemas de transição legendados são modelos básicos que têm problemas em lidar com concorrência e grandes espaços de estados. Apesar dos diagramas de estados e das redes de Petri serem fundamentalmente diferentes, estes partilham as mesmas características. Ambas as técnicas são gráficas, têm semânticas formais e suportam processos concorrentes. Os diagramas de estados centram-se nos estados e nas transições entre estados. As redes de Petri centram-se no fluxo dos objectos (testemunhos) e nas actividades (transições). As álgebras de processos, tais como o ACP, o CSP

e o CCS, não são gráficas e são raramente utilizadas para a modelação de processos de negócio.

Enquanto que o UML reflecte algumas das melhores experiências de modelação disponíveis, este tem uma lacuna, pois não fornece uma semântica precisa, o que é necessário se queremos utilizar a notação para modelar sistemas com precisão e para raciocinar rigorosamente sobre os modelos. Podemos debater que a sintaxe do UML é formalizada. No entanto, em muitas situações a interpretação de uma construção sintáctica é ambígua ou indefinida. O grupo *precise UML* (pUML) quer juntar investigadores internacionais e praticantes que partilhem o objectivo de desenvolver o *Unified Modeling Language* (UML) numa linguagem de modelação precisa (i.e., bem definida). Esta iniciativa mostra que o UML está algures entre uma técnica de modelação de processos formal e informal.

Para concluir, discutimos a relação entre o UML e a técnica de modelação utilizada ao longo deste livro. Existe uma relação clara entre diagramas de actividade e as definições de processos baseadas em redes de Petri utilizadas neste livro. Um diagrama de actividades pode ser traduzido numa rede de Petri transformando actividades em transições, fluxos de objectos em lugares e barras de sincronização em transições. Consequentemente, lugares suplementares necessitam de ser adicionados para interligar as transições. De forma semelhante, é possível fazer uma tradução aproximada das redes de Petri para diagramas de actividades. No diagrama de actividades não existe o conceito de marcas explícitas (i.e., estado global) e o momento de escolha não é bem definido. Por isso, construções subtis, tal como a escolha implícita e as estruturas de escolha não livre, são difíceis de tratar. Os diagramas de interacção, ou seja, os diagramas de sequência e os diagramas de colaboração, podem ser facilmente traduzidos para redes de Petri. Considere, por exemplo, um diagrama de sequência: cada linha de vida é representada por uma sequência de lugares e transições. As mensagens são representadas por lugares que interligam uma transição de uma linha de vida para uma transição de outra linha de vida. A tradução de um diagrama de estados básico para uma rede de Petri é também muito simples: cada estado do diagrama de estados corresponde a um lugar na rede de Petri e cada transição do diagrama de estados corresponde a uma transição na rede de Petri. A tradução de conceitos mais avançados, tal como estados compostos (i.e., agrupamento de estados), regiões ortogonais (i.e., sub-estados concorrentes) e estados históricos, são mais difíceis de traduzir. Da mesma forma, algumas construções das redes de Petri são difíceis de representar utilizando os diagramas de estados (i.e., limitação e escolha não livre). Deve-se notar também que grande parte das técnicas de análise baseadas em diagramas de estados são técnicas *brute-force* que simplesmente exploram o espaço dos estados. Para as redes de Petri, como foi demonstrado no apêndice A, também existem técnicas estruturadas que analisam o processo sem explorar o espaço dos estados.

Note que para cada um dos diagramas mostrados neste apêndice existe uma definição simples de processos baseada em redes de Petri. Isto é deixado como exercício para o leitor interessado.

Soluções do capítulo 1

Exercício 1.1

(a) As regras são:

- *Sequência*: uma depois da outra;
- *Selecção de escolhas*: apenas uma das tarefas será executada, dependendo de uma condição;
- *Paralelismo*: as tarefas podem ser executadas ao mesmo tempo ou numa ordem qualquer; e
- *Iteração*: uma ou mais tarefas devem ser executadas (potencialmente) múltiplas vezes.

(b) A iteração não é uma construção básica: pode ser expressa em termos de "selecção de escolhas."

Exercício 1.2

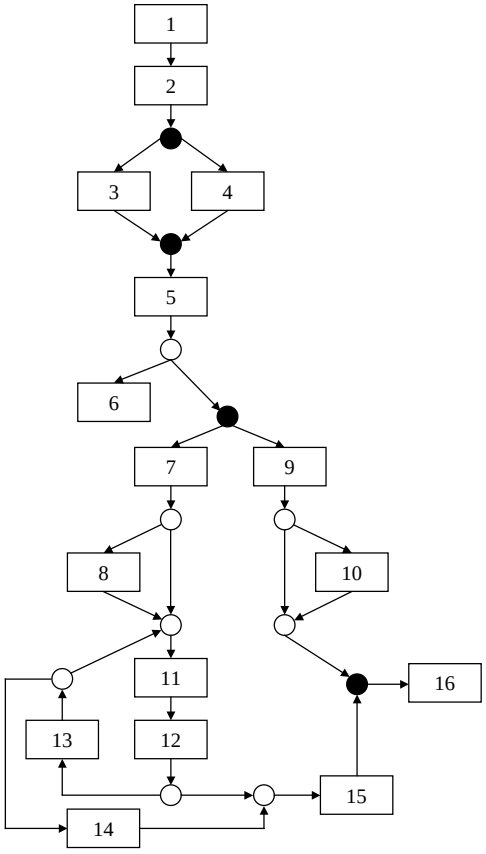


Figure S1.1. O processo de seguro

Exercício 1.3

Considerada dentro de um processo, uma tarefa é uma unidade lógica de trabalho que é executada por um recurso. Considerada do ponto de vista de um (sub) fornecedor, uma tarefa é uma encomenda a ser satisfeita. Entretanto, a satisfação da encomenda pode requerer um processo com diversas tarefas.

Exercício 1.4

Podemos dividir os funcionários em *grupos de capacidade*, em *departamentos funcionais* e em *departamentos de processos ou de produção*. Uma vantagem dos grupos de capacidade é que pessoas com as mesmas competências estão na mesma unidade o que dá uma flexibilidade no planeamento dos recursos. Uma desvantagem é que as unidades não têm nenhuma responsabilidade directa sobre um processo ou gestão de um caso. Uma vantagem dos departamentos de processos é que estão centradas no

desempenho dos processos e na gestão eficiente dos casos. Uma desvantagem é que a troca dos funcionários entre equipas de processo é mais difícil. A organização de um departamento funcional é uma mistura de ambos: não há responsabilidade para a manipulação completa de um caso, mas há responsabilidade para um conjunto de tarefas que pertencem possivelmente a mais do que um processo e que requerem competências similares.

Soluções do capítulo 2

Exercício 2.1 Semáforo Alemão

(a) Os possíveis estados e o sistema de transições são ilustrados na figura S2.1.

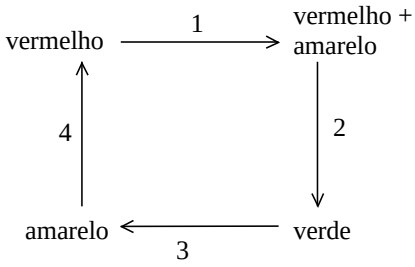


Figura S2.1. Estados e transições

(b) O modelo construído com as linhas contínuas é capaz de se comportar como um semáforo alemão, i.e., ignore os lugares *c1* e *c2*.

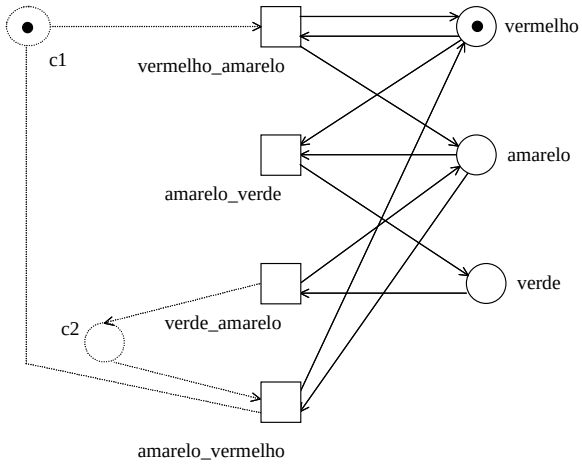


Figura S2.2. Modelo de um semáforo Alemão

(c) A adição dos lugares e arcos ponteados é requerida para fazer o modelo funcionar como um semáforo Alemão. Sem isto, o semáforo *pode* funcionar bem, mas existem potenciais anomalias, tais como:

- A transição *vermelho_amarelo* dispara repetidamente sem transitar para o *amarelo* ou o *verde* e isto resulta na acumulação de testemunhos em *amarelo*.
- O *amarelo_vermelho* pode disparar antes do *verde_amarelo* disparar.

Exercício 2.2 Projecto X

(a)

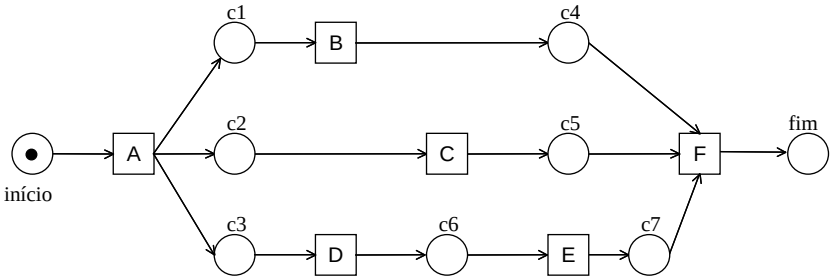


Figura S2.3. O projecto X

(b) Para tornar *E* opcional, tem de ser feito um *bypass* a esta transição

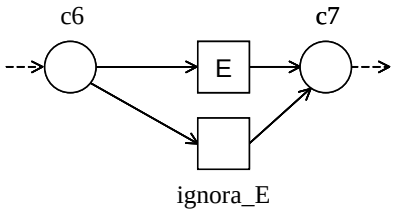


Figura S2.4. Bypass E

(c) O lugar *c8* é introduzido para nos certificarmos que, se a transição *D* é iniciada, *B* e *C* não podem ser executadas porque elas também precisam de um testemunho em *c8*. Quando a transição *E* é finalizada, um testemunho é produzido para *c8* de modo a tornar as novas transições possíveis.

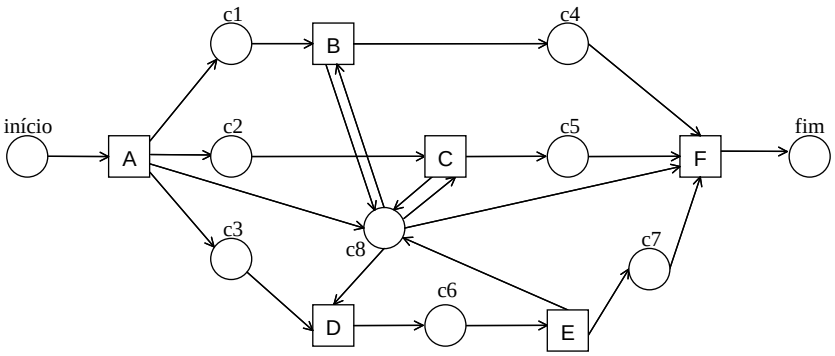


Figura S2.5. A extensão c8

Exercício 2.3 Rede de caminhos-de-ferro

(a) Uma linha pode ser modelada como ilustra a figura S2.6.

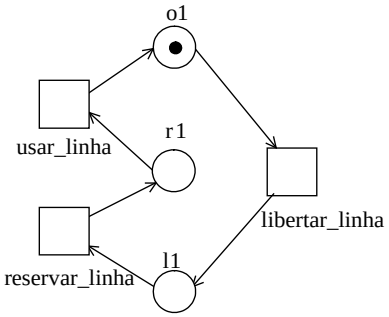


Figura S2.6. Uma linha do caminho-de-ferro

Uma linha consiste em três lugares (*o* = ocupada, *r* = requisitada e *l* = livre) e as transições entre elas. Para criar quatro linhas com dois comboios, copiamos esta linha quatro vezes e colocamos dois testemunhos num lugar *o* e dois testemunhos num lugar *l*.

Temos então de adicionar mais elementos. Um comboio pode mudar de linha apenas se requisitou com sucesso outra linha. Para tal, tem de verificar se a outra linha está livre. Estes são os arcos entre os lugares *o* e a transição *usar_linha*.

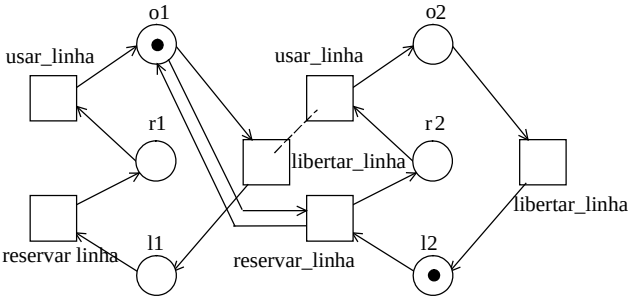


Figura S2.7. Duas linhas de caminho-de-ferro

Note-se também que as transições *usar_linha* e *libertar_linha* de duas linhas subsequentes são executadas ao mesmo tempo. Assim sendo, podemos fundi-las numa só transição: *transferir*.

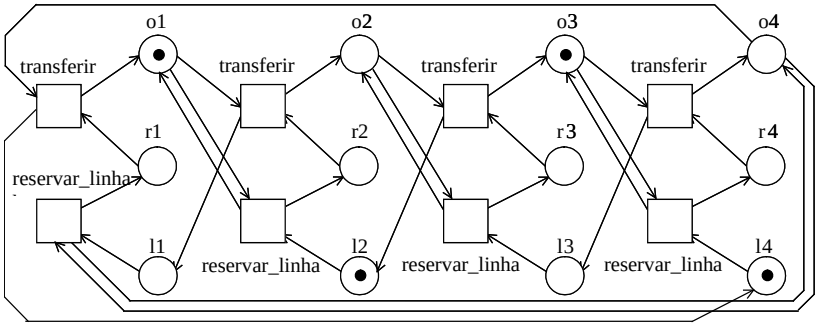


Figura S2.8. O sistema completo constituído por 4 linhas e 2 comboios

(b) Apenas adicione novas linhas. Enquanto o número total de estados aumenta rapidamente, o tamanho da rede de Petri é linear no número de linhas. Note-se que o número de estados é expresso pela seguinte equação:

$$\frac{n * (n-1)}{2} + (n*(n-2)) + \frac{n * (n-3)}{2} = 2n(n-2)$$

Exercício 2.4 Contador binário

Os diferentes estados são os seguintes:

a	b	c		
0	0	0	=	0
0	0	1	=	1
0	1	0	=	2
0	1	1	=	3

a	b	c		
1	0	0	=	4
1	0	1	=	5
1	1	0	=	6
1	1	1	=	7

Obtemos o modelo que podemos visualizar na figura S2.9.

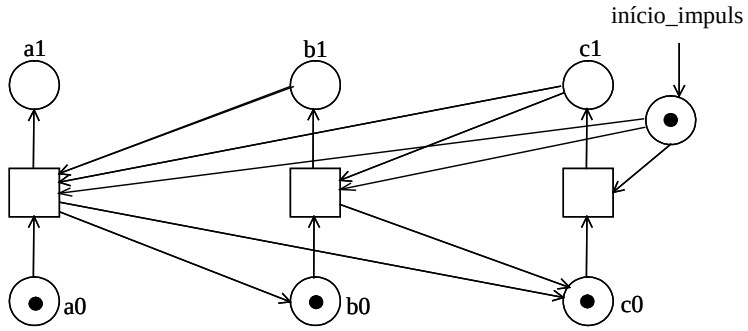


Figura S2.9. O contador binário

Os lugares *a1* e *a0* representam o estado do primeiro dígito, *b1* e *b0* representam o estado do segundo dígito, e *c1* e *c0* representam o estado do terceiro dígito.

Exercício 2.5 Escola de condução

(a)

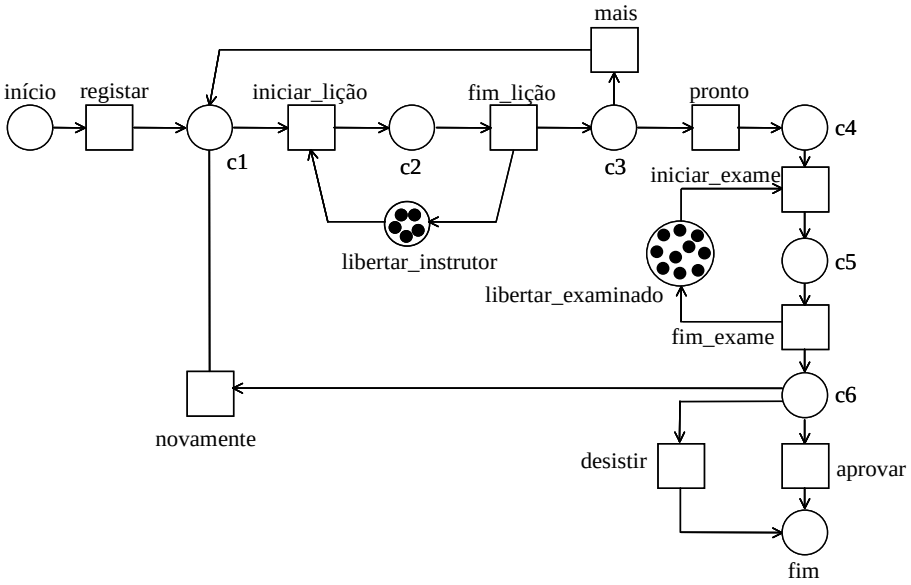


Figura S2.10. A escola de condução

(b) Todos os testemunhos nos lugares *início, c1*, *c2*, *c3*, *c4*, *c5*, *c6*, *fim* têm agora um valor. Por exemplo: uma pessoa chamada J. Walker, com 18 anos de idade que ainda não teve lições ou exames é representada da seguinte forma:

[id: ‘X07’; nome: ‘J. Walker’; idade: ‘18’; sexo: ‘masculino’; num_lições: ‘0’; num_exames: ‘0’]

Os últimos dois atributos são importantes para o exercício, pois queremos saber quantas lições e exames uma pessoa já teve. As transições são especificadas da seguinte forma:

registar: num_lições: = 0
num_exames: = 0

A transição para *mais* e *pronto* podem ser fundidas numa só transição: *mais?* com o seguinte comportamento:

```
if
  num_lições < 10
then
  produz testemunho para c1
else
  produz testemunho para c4
```

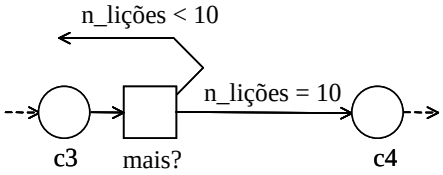


Figura S2.11. As transições *mais* e *pronto* combinadas na transição *mais?*

fim_lição: num_lições: = num_lições + 1

fim_exame: num_exames: = num_exames + 1

novamente tem uma pré condição: num_exames < 3

Colocamos o atributo num_lições: = 0, porque temos de ter outras 10 lições antes de outro exame.

(c) Todos os atrasos são iguais a zero excepto o indicado na figura S2.12.

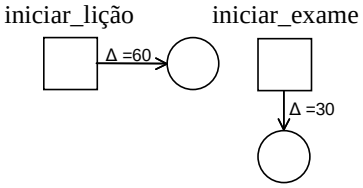


Figura S2.12. Adição de atrasos positivos

Exercício 2.6 Fábrica de bicicletas

(a)

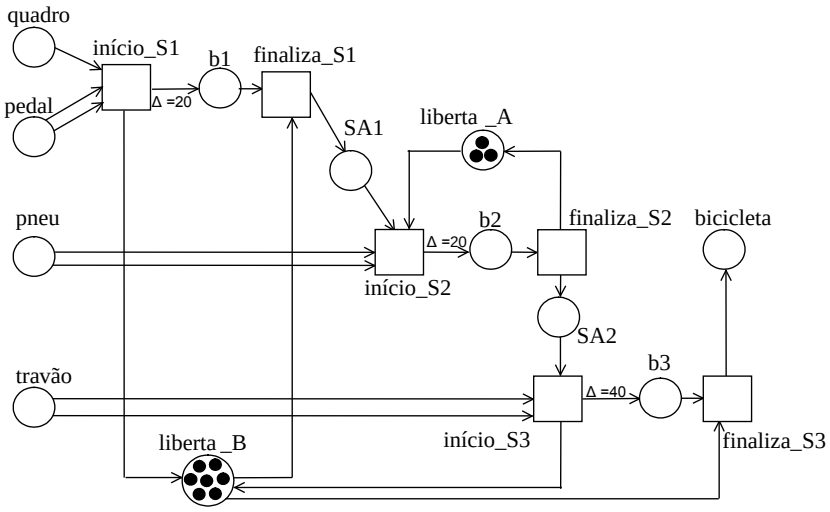


Figura S2.13. A fábrica de bicicletas

(b) Capacidade A: 3 * (60 minutos/20 minutos de trabalho SA2) = 9 b/h
Capacidade B: 7 * (60 minutos/(20 + 40 minutos de trabalho SA1 e SA3)) = 7 b/h

Identificamos a capacidade da máquina B como sendo aquela que origina um congestionamento e, por isso, a capacidade da fábrica para produzir é de sete bicicletas por hora.

Exercício 2.7 Companhia de seguros

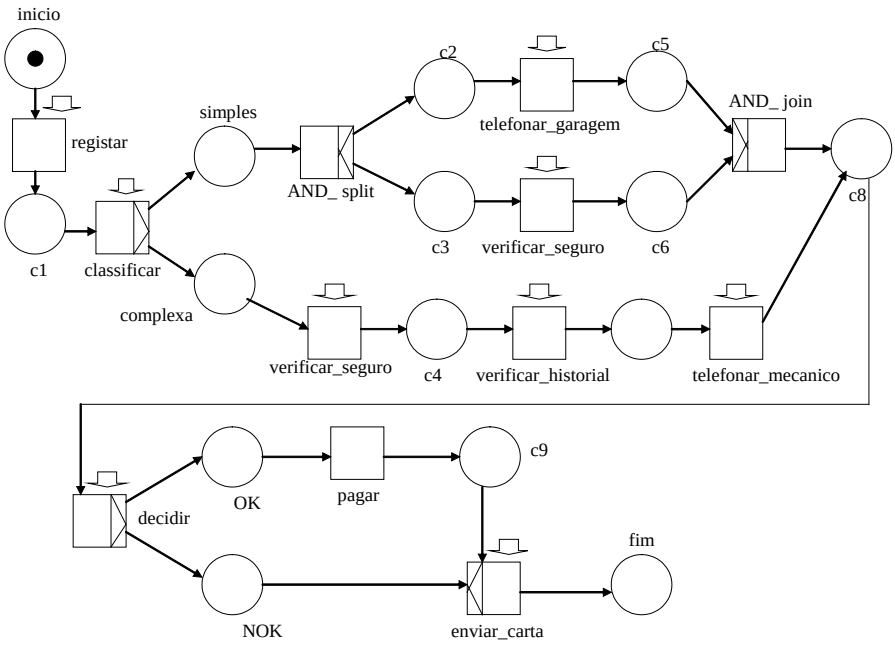


Figura S2.14. A companhia de seguros.

A escolha entre *OK* (e depois *pagar*) e *NOK* pode também ser feita com um lugar para os lugares *NOK* e *c9*. Neste caso, *enviar_carta* só tem um lugar de entrada e não é requerida a notação OR_join.

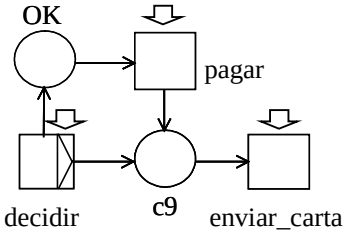


Figura S2.15. Remoção do OR-join por fusão dos lugares c9 e NOK

Exercício 2.8 Gestão de reclamações

A parte mais difícil de modelar é a relação entre o manuseamento do formulário e o processamento propriamente dito: a tarefa *processar* tem de esperar até que o manuseamento do formulário seja finalizado e pode ser executada um número arbitrário de vezes. Na figura S2.16, este problema é resolvido com o auxílio de duas tarefas para o processamento propriamente dito: *processar* e *processar_novamente*. Na figura S2.17, só existe uma tarefa chamada *processar*. Aqui, a tarefa *processar* remove um testemunho de *c9*, mas também volta a colocar um imediatamente a seguir. Como resultado, a tarefa *processar* pode ser executada um número arbitrário de vezes, sem remover o testemunho de *c9*.

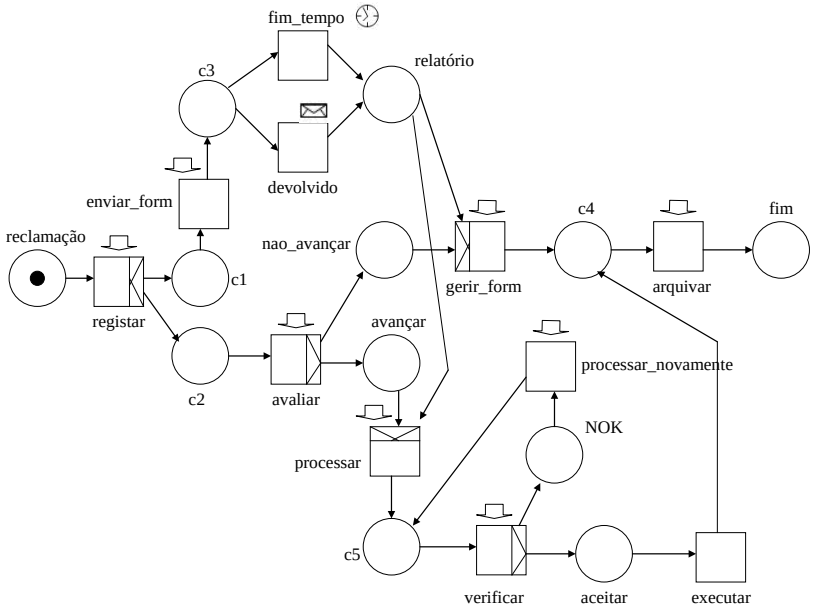


Figura S2.16. Gestão de reclamações

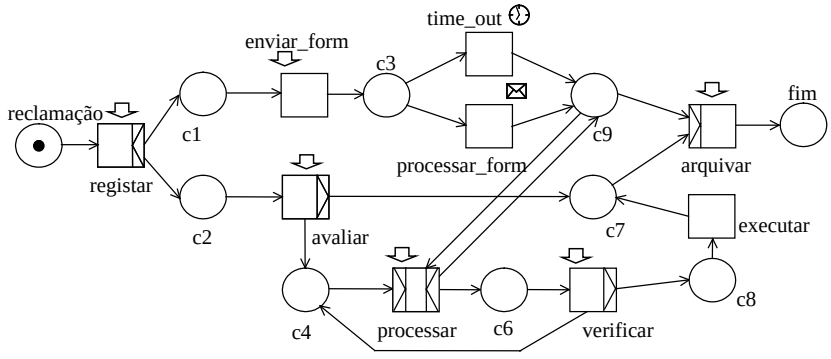


Figura S2.17. Gestão de reclamações

Exercício 2.9 Vamos dar uma festa

(a) Podem ser identificadas três partes do processo:

- organizar o local
- organizar a música
- considerações finais (conta, comida, bebidas e visita)

As primeiras duas partes são executadas em paralelo, seguidas da terceira parte. A segunda parte (música) é a parte mais complexa do processo. Dois OR-splits implícitos são necessários para manipular os *time-outs*.

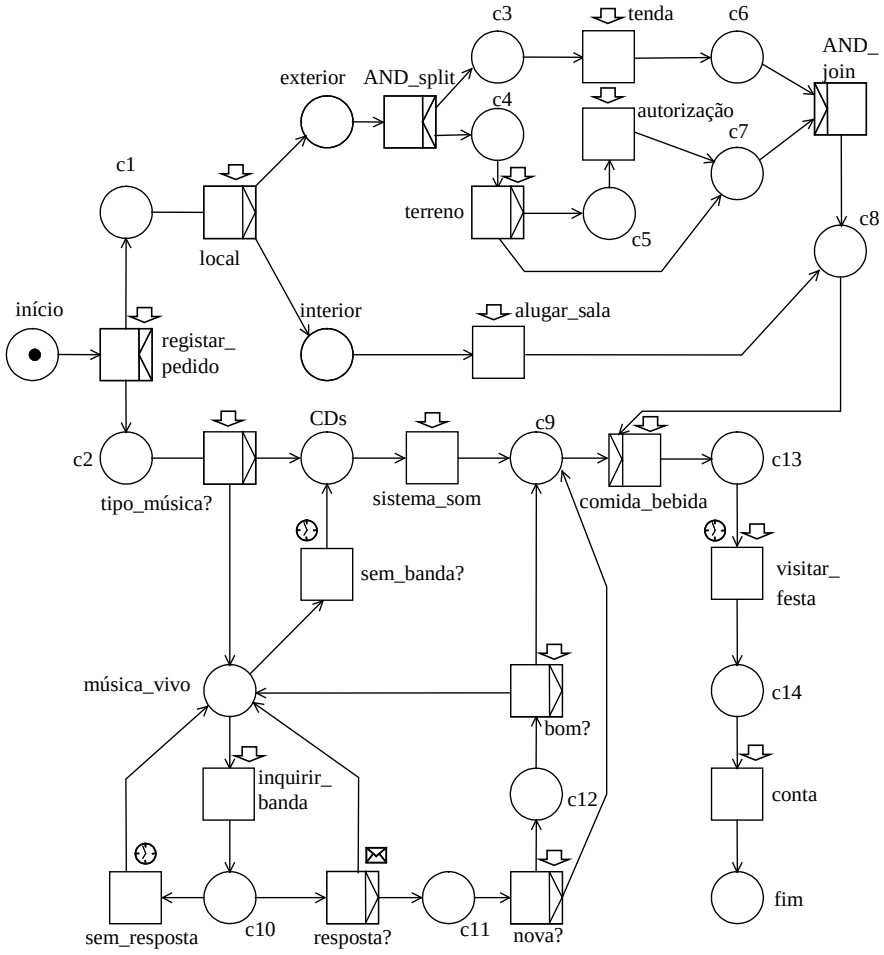


Figura S2.18. Festa

(b) Melhoramentos:

O mais importante congestionamento no processo é a selecção da banda. Esta parte do processo pode levar muito tempo, particularmente quando uma ou

duas bandas recusam, ou quando o desempenho de uma banda é muito fraco. Para tal, a melhoria mais significativa pode ser obtida quando o processo é separado em dois: um para manusear os pedidos para festas e outro para avaliar as bandas. Como resultado, as bandas podem ser avaliadas independentemente dos requisitos específicos para festas.

Soluções do capítulo 3

Exercício 3.1 Companhia de Seguros

Foram identificados os seguintes papéis:

Funcionário	(F)
Gestor Declarações	(GD)
Gestor Declarações A	(GDA)
Gestor Declarações B	(GDB)

Foram identificadas as seguintes unidades organizacionais:

Departamento Danos Automóvel	(DDA)
Departamento Financeiro	(DF)

Estes resultados estão descritos na figura S3.1.

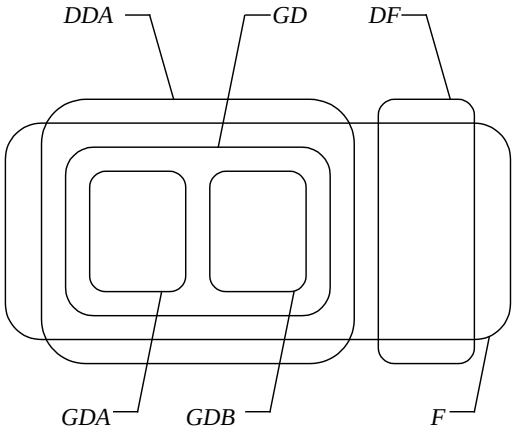


Figura S3.1. A classificação dos recursos da companhia de seguros

Assumimos que todos os gestores de declarações também são funcionários. Isto significa que quando é necessário um funcionário do departamento de Danos Automóvel para executar uma tarefa, não é importante se ele é ou não um gestor de declarações. Se assumirmos que um gestor de declarações não

pode efectuar a tarefa de um funcionário "normal", então a figura S3.1 necessita de ser adaptada (GD, GDA e GDB estarão fora de F).

Se combinarmos as classificações de recursos com o modelo de processo, obtemos o modelo ilustrado na figura S3.2.

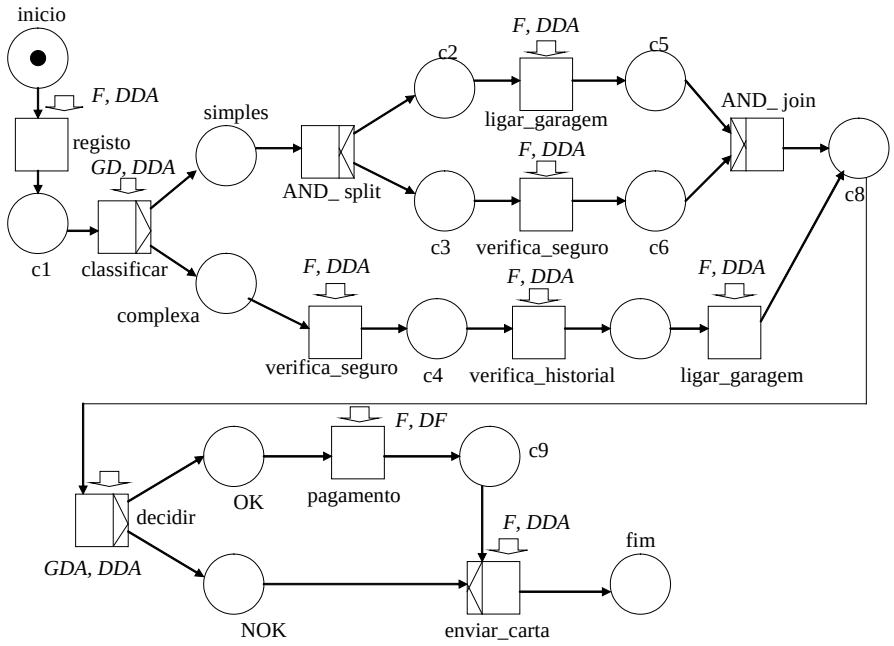


Figura S3.2. Classificação dos recursos no modelo da companhia de seguros

Exercício 3.2 Gestão de reclamações

Foram identificados os seguintes papéis:

Funcionário	(F)
Gestor de reclamações	(GR)

Foram identificadas as seguintes unidades organizacionais:

Departamento C	(DC)
Departamento de logística	(DL)

Estes resultados são ilustrados na figura S3.3.

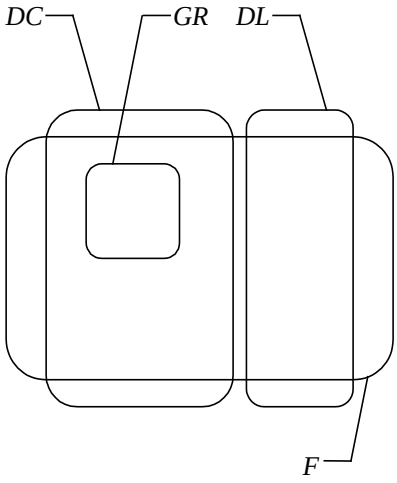


Figura S3.3. Classificação dos recursos da gestão de reclamações

Neste exercício também assumimos que o gestor de reclamações é um funcionário. Isto significa que ele também está disponível para trabalho que poderia ser feito por um funcionário.

Se combinarmos a classificação de recursos com o modelo do processo, obtemos o modelo ilustrado na figura S3.4.

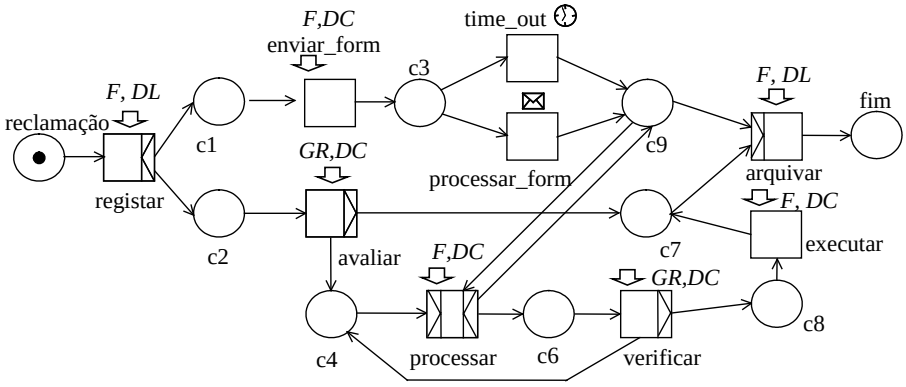


Figura S3.4. Classificação dos recursos do modelo de gestão de reclamações

Exercício 3.3 Agência de emprego

(a) Foram identificados os seguintes papéis:

Relações Públicas	(RP)
Relações de Empresariais	(RE)
Recrutamento	(RC)

Gestor	(GE)
Especialista IT	(IT)

Foram identificadas as seguintes unidades organizacionais:

Loja de Emprego	(LE)
Eindhoven	(EH)
Leeuwarden	(LW)

Os resultados são ilustrados na figura S3.5.

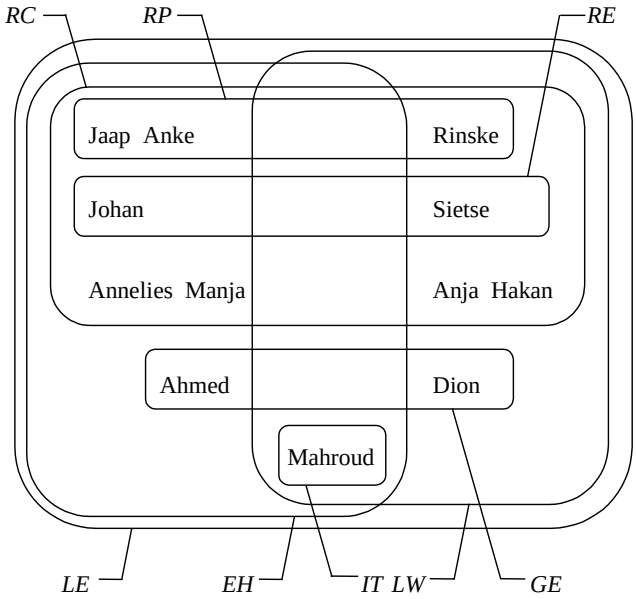


Figura S3.5. Classificação dos recursos da agência de emprego

(b) A figura S3.6 ilustra o modelo do processo. É importante colocar os disparos correctos. Por exemplo, o disparo tempo adicionado à tarefa *parar_processamento* é crucial para manter o fluxo e prevenir que casos fiquem presos no lugar *aguardar*.

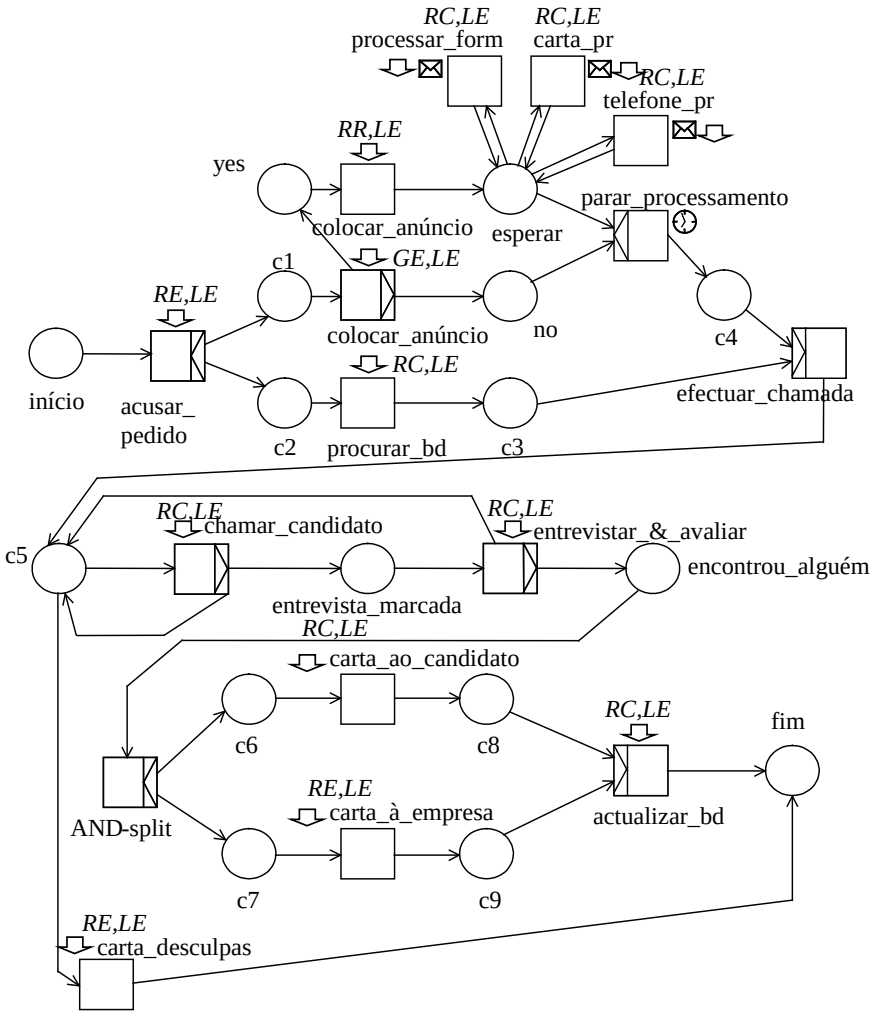


Figura S3.6. O processo da agência de emprego

Exercício 3.4 Tenha um bom voo com CRASH

(a) Foram identificados os seguintes papéis:

Loadmaster	(LM)
Navegador	(NV)
Capitão	(CP)
Meteorologia	(MT)
Director	(DR)
Logística	(LG)
Secretária	(SE)

Mensageiro (ME)

Foram identificadas as seguintes unidades organizacionais:

AIR (AR)
KLM (KL)
Apoio (AP)
CRASH (CR)

Os resultados são mostrados na figura S3.7.

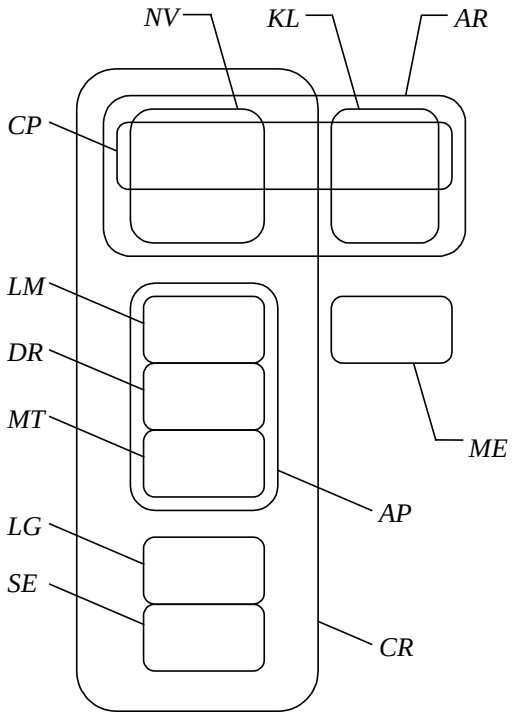


Figura S3.7. Os recursos da empresa CRASH

(b) O processo é simples; aplique simplesmente os construtores de encaminhamento básicos. A tarefa *discussão* requer dois recursos: um navegador e um *loadmaster*. Logo, dois papéis estão anexados a esta tarefa: *NV* e *LM* (ver a figura S3.8). Porque eles são ambos membros de *CR*, usamos a notação *NV/LM*, *CR*. É também possível considerá-los membros independentes de uma unidade organizacional diferente e neste caso usa-se a notação *NV*, *AR/LM*, *CR*. Este conceito é também utilizado nas outras tarefas onde são necessários dois recursos diferentes. Note-se que a actual geração dos sistemas de *workflow* não suporta a especificação de recursos múltiplos a trabalhar num item de trabalho. Portanto, evitamos tarefas com recursos múltiplos sempre que possível.

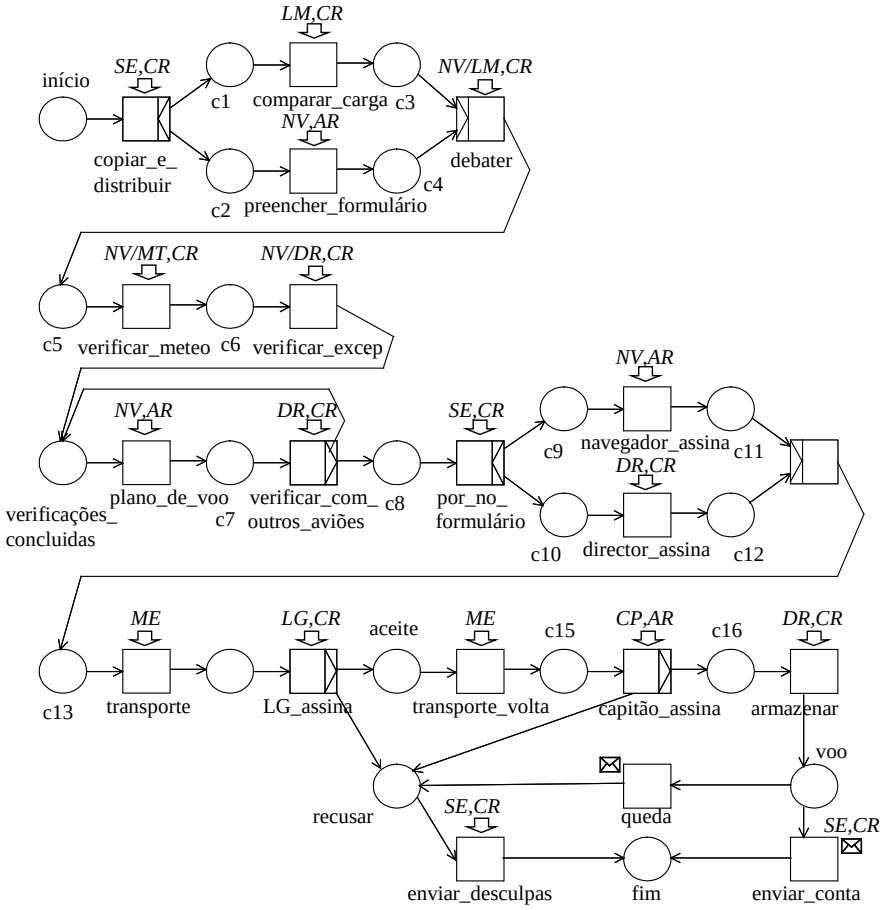


Figura S3.8. O processo *CRASH*

(c) Possíveis melhoramentos: a introdução de documentos electrónicos (sistema de *workflow*) pode melhorar o tempo de execução dos processos. Várias tarefas tornam-se redundantes (e.g. *copiar_e_distribuir*, *por_no_formulário*) e o nível de paralelismo pode ser aumentado. Mais ainda, as tarefas *LG_assina* e *capitão_assina* devem ser executadas o mais cedo possível para evitar trabalho para voos que nunca chegam a ser efectuados.

Soluções do capítulo 4

Exercício 4.1 Optimização do uso de dados

(a)

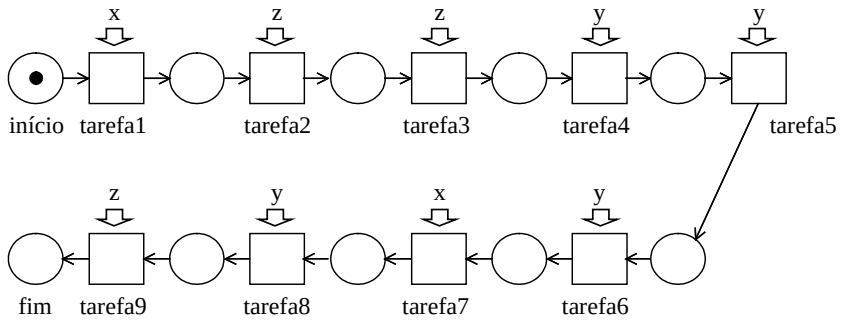


Figura S4.1. Um processo sequencial

(b) Não, não é possível representar várias formas de encaminhamento tais como o encaminhamento selectivo e paralelo.
(c) Na figura S4.2 podemos ver todas as relações de precedência. Na figura S4.3 ignoramos aquelas que podem ser derivadas, isto é, se a *tarefa1* tem de ser executada antes da *tarefa2* e da *tarefa7*, e a *tarefa2* também tem de ser executada antes da *tarefa7*, a relação entre a *tarefa1* e a *tarefa7* pode ser derivada e por isso omitida. Isto resultará na rede de Petri apresentada na figura S4.4.

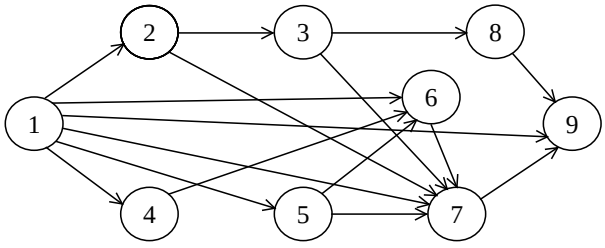


Figura S4.2. O processo completo

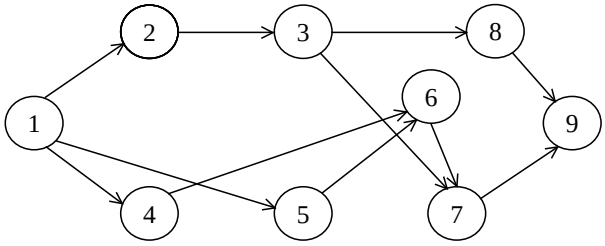


Figura S4.3. O processo parcial

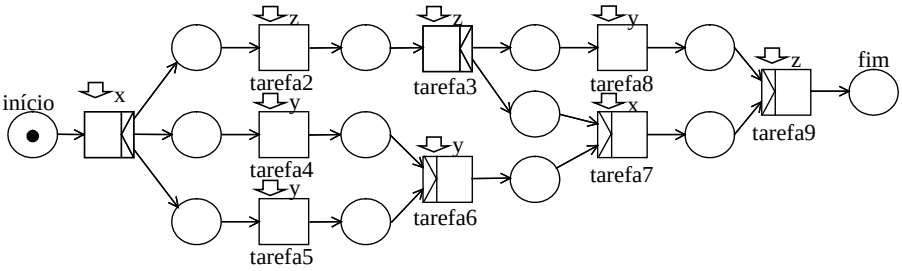


Figura S4.4. Rede de Petri

(d) Sim. As tarefas 2 e 3 e as tarefas 4, 5 e 6 são executadas por um tipo de recurso e por isso podem ser agrupadas. Consequentemente, podem ser combinados numa só tarefa.

Exercício 4.2 Invariantes

i) Primeira rede de Petri (figura 4.38)
a) descansar_e + escrever_mail (=1)
descansar_l + ler_mail (=1)
b) início + escrever_mail + receber_mail + ler
c) Não, pode existir um número arbitrário de testemunhos no lugar *mailbox*.
d) Sim.
e) Sim.
f) {descansar_e, escrever_mail, início, escrever_mail}, {ler_mail, descansar_l, receber_mail, ler}

ii) Segunda rede de Petri (figura 4.39)
a) *c1* + *c2* (=1)
c3 + *c4* (=1)
b) *a* + *b* + *c* + *d*
c) Sim.
d) Sim.
e) Não.
f) {*c1*, *c2*, *a*, *b*, *c*, *d*}, {*c3*, *c4*, *a*, *b*, *c*, *d*}

iii) A terceira rede de Petri (figura 4.40)

a) *c1 + c4* (=1)
c2 + c5 (=1)
c3 + c6 (=1)

b) *g*
a + b
c + d
e + f
c) Sim.
d) Sim.
e) Não.
f) {*c1*, *c4*, *a*, *b*, *g*, *e*}, {*c2*, *c5*, *a*, *c*, *d*, *g*}, {*c3*, *c6*, *e*, *f*, *c*, *g*}

iv) A quarta rede de Petri (figura 4.41)

a) *início + c1+ c2 + c3 + c4 + fim* (=1)
início + encomendar_a + c5 + c7 + c9 + c11 + c13 + factura + c4 + fim (=1)
início + encomendar_a + c6 + c8 + notificação + c2 + c3 + c4 + fim (=1)
c5 + c7- c6 - c8 (=0)
c9 + c11 - cl0 - c12 (=0)
Etc.
b) produzir_b + verificar_b + NOK_b
produzir_c + verificar_c + NOK_c
c) Sim.
d) Não.
e) Sim.
f) Nenhum.

Exercício 4.3 Verificação da definição de processos

a)

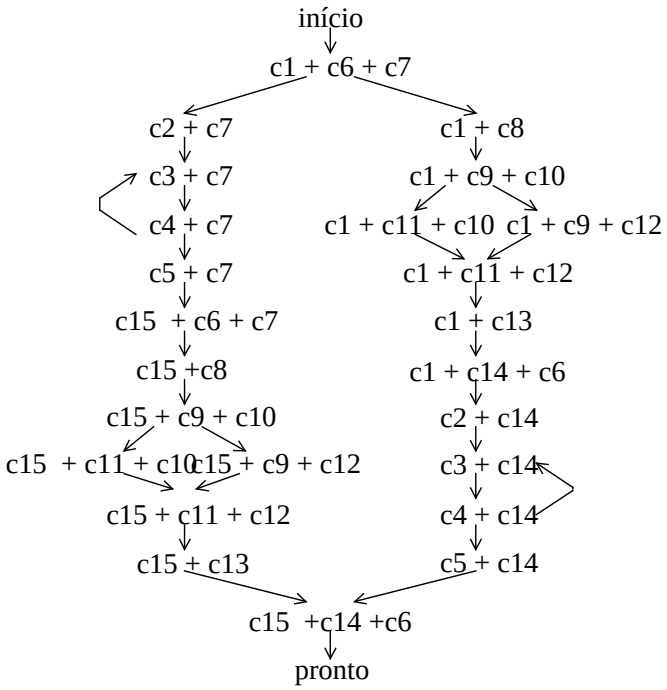


Figura S4.5. Grafo de alcançabilidade

b) 1 + (6 * 8) + 1 = 50. Note que o processo original tem apenas 26 estados.
c) *início* + *c6* + *c2* + *c3* + *c4* + *c5* + *c8* + ½ (*c9* + *c10* + *c11* + *c12*) + *c13* + *pronto*
d) *início* + *c1* + *c2* + *c3* + *c4* + *c5* + *c15* + *pronto*
início + c7 + c8 + c9 + c11 + c13 + c14 + pronto
início + c7 + c8 + c9 + c10 + c12 + c14 + pronto
início + ½ (*c1* + *c2* + *c3* + *c4* + *c5* + *c15*) + ½ (*c7* + *c8* + ½ (*c9* + *c10* + *c11* + *c12*) + *c13* + *c14*) + *pronto*

Exercício 4.4 Procura de erros

i) Se um formulário é processado e *avaliar* produz um testemunho em *c7*, um testemunho ficará em *c9*. Quando ocorre um *time_out* e *avaliar* produz um testemunho em *c4*, o processo fica em *deadlock* no estado marcado por *c8* e *c4*.
ii) Porque *c9* começa num lugar vazio e continua vazio, o processo não pode continuar quando os testemunhos são colocados em *c1* e *c2*.

iii) Se a parte superior do processo alcança c8 antes de um testemunho na parte inferior do processo alcançar o *c4*, o *processo* é incapaz de disparar e paralisa (*deadlock*) no estado marcado por *c8* e *c4*.

Exercício 4.5 Analise de desempenho I

Usamos as seguintes fórmulas:

$$L = \frac{p}{1-p} \qquad S = \frac{1}{\mu - \lambda} \qquad W = \frac{p}{\mu - \lambda}$$

a) Tarefa 1:

$\lambda = 20$
$\mu = 60/2 = 30$
$\rho = 0{,}67$
$L = 2$
$S = 0{,}1$ (6 minutos)
$W = 0{,}066$ (4 minutos)

Tarefa 2:

$\lambda = 20$
$\mu = 60/2.5 = 24$
$\rho = 0{,}83$
$L = 5$
$S = 0{,}25$ (15 minutos)
$W = 0{,}208$ (12,5 minutos)

Total:

$L^T = 7W^T = 0{,}274$ (16,5 minutos)
$S^T = 0{,}35$ (21 minutos)

b) Tarefa 1:

$\lambda = 20$
$\mu = 60/2 = 30$
$\rho = 0{,}67$
$L = 2$
$S = 0{,}1$ (6 minutos)
$W = 0{,}066$ (4 minutos)

Tarefa 2:

$\lambda = 5$
$\mu = 10$
$\rho = 0{,}5$
$L = 1$
$S = 0{,}2$ (12 minutos)
$W = 0{,}1$ (6 minutos)

Total:

$L^T = 3$
$S^T = 0{,}1 + 1/4 * 0{,}2 = 0{,}15$ (9 minutos) $\Delta = -12$ min., i.e., 12 minutos a menos que anteriormente.

Exercício 4.6 Análise de desempenho II

a) Tarefa 1a:

$\lambda = 10$ $\rho = 0{,}8333$ $S = 0{,}5$

$\mu = 12$ $L = 5$ $W = 0{,}04167$

Tarefa 1b:

$\lambda = 10$ $\rho = 0{,}33$ $S = 0{,}05$

$\mu = 30$ $L = 0{,}5$ $W = 0{,}0166$

Tarefa 2:

$\lambda = 20$ $\rho = 0{,}66$ $S = 0{,}100$

$\mu = 30$ $L = 2$ $W = 0{,}066$

Total:

$L^T = 5 + 0{,}5 + 2 = 7{,}5$

$S^T = 1/2 * 0{,}5 + 1/2 * 0{,}05 + 0{,}100 = 0{,}375$ (22,5 minutos)

b) Tarefa 1:

$\lambda = 20$ $\rho = 0{,}66$ $S = 0{,}100$

$\mu = 30$ $L = 2$ $W = 0{,}066$

Tarefa 2:

$\lambda = 20$ $\rho = 0{,}66$ $S = 0{,}100$

$\mu = 30$ $L = 2$ $W = 0{,}066$

Total:

$L^T = 2 + 2 = 4$

$S^T = 0{,}1 + 0{,}1 = 0{,}2$ (12 minutos) $\Delta = -10{,}5$ min.

Exercício 4.7 Análise de desempenho III

a) ct1: (% = 1,0)

$\lambda = 10$ $\rho = 0{,}833$ $S = 0{,}5$

$\mu = 12$ $L = 5$ $W = 0{,}04167$

ct2: (% = 0,8)

$\lambda = 8$ $\rho = 0{,}533$ $S = 0{,}143$

$\mu = 15$ $L = 1{,}14$ $W = 0{,}076$

bt: (% = 0,56)

$\lambda = 5{,}6$ $\rho = 0{,}28$ $S = 0{,}0694$

$\mu = 20$ $L = 0{,}389$ $W = 0{,}0194$

Total:

$L^T = 6{,}53$

$S^T = 1*0{,}5 + 0{,}8*0{,}143 + 0{,}56*0{,}0694 = 0{,}5 + 0{,}114 + 0{,}0389 = 0{,}65$ (39,2 min).

(b) Alternativa 1:

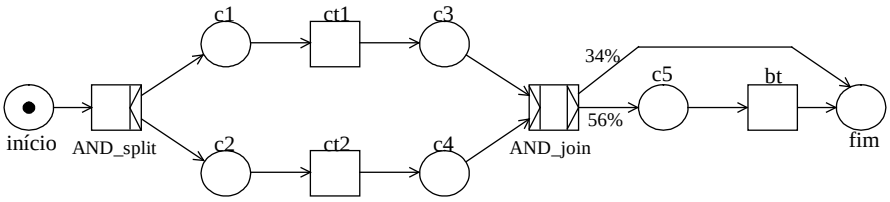


Figura S4.6. A alternativa 1

É possível reduzir o tempo de fluxo executando tarefas em paralelo.

ct1: (% = 1,0)

$\lambda = 10$ $\rho = 0{,}833$ $S = 0{,}5$

$\mu = 12$ $L = 5$

ct2: (% = 1,0)

$\lambda = 10$ $\rho = 0{,}67$ $S = 0{,}2$

$\mu = 15$ $L = 2$

bt: (% = 0,56)

$\lambda = 5{,}6$ $\rho = 0{,}28$ $S = 0{,}07$

$\mu = 20$ $L = 0{,}389$

O *ct1* provoca um congestionamento (*bottleneck*) no processo paralelo.

Total:

$L^T > 5{,}39$

Saída (*throughput*) máxima = λ * (1/ $\rho_{\text{bottleneck}}$) = 10 * (1/0,833) = 12

Alternativa 2:

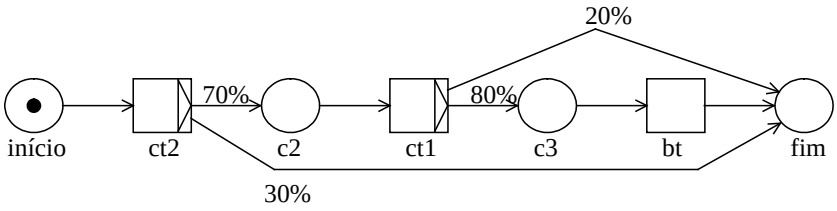


Figura S4.7. A alternativa 2

Neste caso, mais testemunhos irão directamente para o lugar *fim*, assim os recursos serão menos usados.

ct1: (% = 0,7)

$\lambda = 7$ $\rho = 0{,}583$

$\mu = 12$ $L = 1{,}4$

ct2: (% = 1,0)

$\lambda = 10$ $\rho = 0{,}67$

$\mu = 15$ $L = 2$

bt: (% = 0,56)

$\lambda = 5{,}6$ $\rho = 0{,}28$

$\mu = 20$ $L = 0{,}389$

O *ct2* tornou-se agora no ponto de congestionamento e existem poucos casos no sistema.

Total:

$L^T = 3{,}79$

Saída (*throughput*) máxima = λ * (1/ $\rho_{\text{bottleneck}}$) = 10 * (1/0,67) = 15

Outras Alternativas:

- Combinar *ct1* e *ct2* numa única tarefa para reduzir o tempo de preparação (*setup*).
- Disponibilizar um conjunto de recursos únicos para todas as tarefas.

Exercício 4.8 E-business

(a)

passo	conjunto de tarefas	tarefa seleccionada	bloco usado	nova tarefa
1	a	a	sequência	b
2	a,b	b	sequência	c
3	a,b.c	b	iteração	d

workflow do cliente

passo	conjunto de tarefas	tarefa seleccionada	bloco usado	nova tarefa
1	e	E	sequência	f

workflow do servidor

(b)

passo	conjunto de tarefas	tarefa seleccionada	bloco usado	nova tarefa
1	a	a	sequência	b
2	a,b	b	sequência	c
3	a,b.c	b	iteração	d
4	a,b,c,d	b	and	e
5	a,b,c,d,e	e	sequência	f

workflow acoplado

Os passos 1, 2 e 3 do *workflow* acoplado são os mesmos para o *workflow* do cliente. O passo 4 é novo e o passo 5 é o passo 1 do *workflow* do servidor.

(c) Não, tal derivação não é possível. Para verificar isso note-se que (p,q), (t,v) e (r,s) formam pares de AND-splits e AND-joins. Assim, cada um deles deve ser feito por uma substituição de um bloco AND. Porém, eles seriam aninhados (um incluído no outro) ou disjuntos. Não é este o caso; na realidade têm a seguinte sequência: *p, t, q, r, v, s*. Assim, eles cruzam-se entre si.

(d) Sim, é um *workflow* seguro e correcto. Para ver isso note-se que sem a troca de mensagens, isto é, sem *q, t, r* e *s* temos um *workflow* seguro e correcto (ver o exercício 1). Como *b* e *d* irão disparar, podemos observar que *t* e depois *q* irão disparar, assim como também *c* e *e*. Semelhantemente, *r* e *v* irão disparar posteriormente. Assim, *f* e mais tarde *s* também irão disparar. Porque não resta nenhum testemunho na rede, ela é correcta. O facto da rede ser segura é uma consequência directa da rede sem troca de mensagem ser também segura.

Soluções do capítulo 5

Exercício 5.1

A figura S5.1 ilustra uma representação gráfica do modelo de referência do WFMC. Para uma descrição mais detalhada dos componentes e das interrelações consultar o capítulo 5.

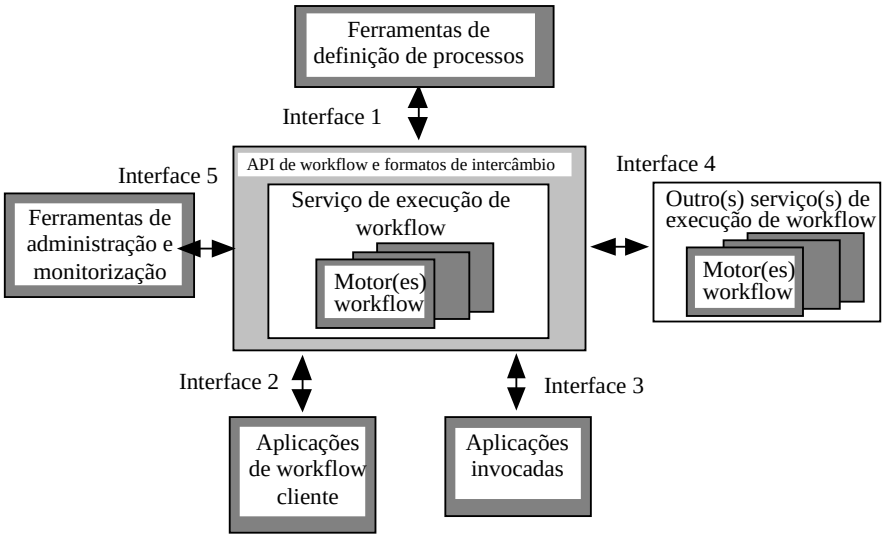


Figura S5.1. O modelo de referência do Workflow Management Coalition (© WFMC)

Exercício 5.2

Respostas às perguntas:

(a) Atomicidade, Consistência, Isolamento e Durabilidade.
(b) Interface 3: O sistema de gestão de workflows e as aplicações não estão sincronizadas.
(c) Designer de workflows, Administrador, Analista de processos e Funcionários.
(d) Interoperabilidade: Especificação do WFMC, SWAP, WF-XML e do jointFlow do OMG.
(e) Staffware: é um dos líderes de mercado centrado no desenvolvimento de workflows de produção. COSA: é um sistema de gestão de *workflow* baseado em redes de Petri também orientado para workflows de produção. ActionWorkflow: é um sistema que fomenta a colaboração e negociação em detrimento do encaminhamento e é completamente diferente dos sistemas de produção típicos.
(f) Woflan: é uma ferramenta de verificação que usa técnicas avançadas de análise, isto é, análise qualitativa. ExSpect: é uma ferramenta de simulação

baseada em redes de Petri. Ambas as ferramentas podem ser usadas em combinação com diversos produtos de *workflow*.
(g) Protos (Pallas Athena BV, Plasmolen, The Netherlands), ARIS (IDS Scheer AG, Saarbrücken, Germany), BusinessSpecs (IvyTeam, Zug, Switzerland), Income (Promatis AG, Karlsbad, Germany) e Meta WorkflowAnalyzer (Meta Software, Cambridge, MA, USA).

Exercício 5.3

O COSA é baseado em redes de Petri. Por isso, existe uma tradução de um para um e por isso não ilustramos esse procedimento usando o CONE. A tradução do processo para o sistema Staffware é mais complicada. A figura S5.2 mostra a definição de um processo de *workflow* no GWD do Staffware. O modelo é simples, ilustrando os blocos construtores usados.

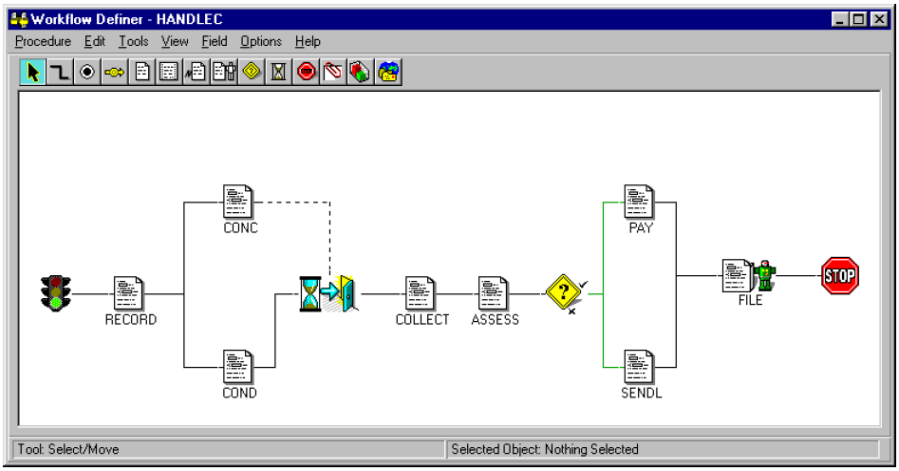


Figura S5.2. Processo "tratar reclamação" modelado usando o GWD da Staffware

Exercício 5.4

Existe uma tradução de um para um do modelo ilustrado no capítulo 2 para o COSA. A tradução do processo para Staffware é mais complicada. A figura S5.3 ilustra a definição de um processo de *workflow* no GWD do Staffware. A primeira parte do modelo é simples, dando uma descrição dos blocos construtores usados. O único elemento que é menos trivial de modelar é a tarefa de cancelamento. Tipicamente, as construções que não são de escolha livre são difíceis, se não impossíveis, de modelar usando o Staffware. Nestes casos podemos usar um artifício simples para a sua modelação: duas etapas de cancelamento com um intervalo de tempo. Por questões de simplicidade, resolvemos não modelar os disparos e simplificámos a escolha para ambos os tipos de seguros.

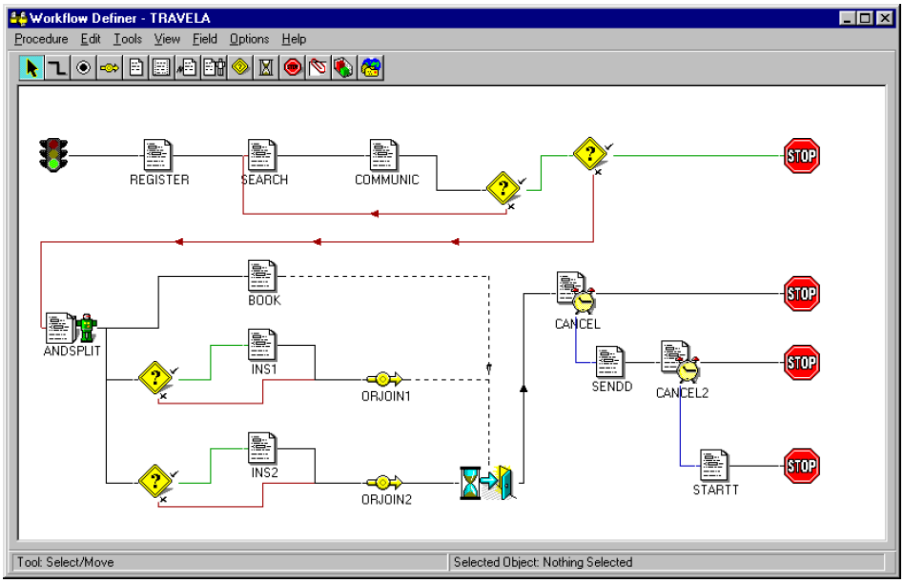


Figura S5.3. O processo "agência de viagens" modelado usando o GWD da Staffware

Soluções do capítulo 6

Exercício 6.1

(a) Primeiro, é importante envolver (potenciais) utilizadores porque têm um conhecimento vasto e aprofundado sobre os processos e os sistemas existentes. Normalmente também têm sugestões interessantes para realizar melhoramentos. Por consequência, o seu conhecimento e a sua criatividade são uma mais valia para a equipa de redesenho.

Segundo, o seu envolvimento é importante para obter um compromisso dentro da organização. As pessoas que participam activamente na fase de desenho de novos processos e sistemas têm um "carinho" especial por estes. Logo, estão dispostas a defender os novos processos e sistemas da influência de outras pessoas, em particular dos seus colegas. Então, passam a ser elementos chave no processo de mudança. Isto é essencial porque muito frequentemente as operações de mudança encontram alguma resistência por parte dos funcionários que não fazem parte da equipa. O processo de mudança é muito emocional.

(b) É muito importante seleccionar pessoas com as seguintes características:

- Respeitados pelos colegas
- Conhecedores dos processos ou sistemas

- Terem uma mente aberta. Por exemplo, possuírem a capacidade "think outside the box"

Exercício 6.2

Na *fase de diagnóstico*, os casos de negócio são usados para determinar os valores dos *Key Performance Indicators* (KPI, indicadores chave de desempenho) na situação actual: a medida nula. É por vezes mais fácil explicar algo usando um exemplo do que formular uma regra que descreve o exemplo. Casos de negócio podem ser considerados como exemplos, enquanto que os processos constituem as regras. Para os utilizadores é mais fácil "pensarem" em casos de negócio do que em termos de processos pois estes são mais abstractos. A fase seguinte, na qual são utilizados, é a *fase de redesenho dos processos*, isto é, são usados para realizar experiências de simulação e jogos. Também podem ser utilizados na especificação dos requisitos. Finalmente, os casos de negócio são também usados na *fase de integração*, quando o sistema é testado e na *fase de entrega*, quando é realizado o teste de aceitação. Em seguida, é importante manter cuidadosamente o conjunto de casos de utilização. Desta forma podem também ser utilizados durante a *fase de monitorização e melhorias*, isto é, quando são consideradas melhorias.

Exercício 6.3

As vantagens de combinar as fases são as que se seguem: é vantajoso especificar um modelo conceptual de dados e um modelo funcional em conjunto com a estrutura dos componentes, porque a distribuição de funcionalidades pelos componentes pode ser derivada de uma forma iterativa. Se os modelos de requisitos e a arquitectura são divididos em duas fases, a iteração é mais difícil. Pode ser uma vantagem considerar os detalhes técnicos e funcionais numa só fase, porque previne a definição de requisitos tecnicamente impossíveis de satisfazer.

Existem também desvantagens. Na fase de requisitos os utilizadores podem dar um contributo significativo, enquanto estes são menos úteis na fase da arquitectura técnica. Portanto, é natural dividir as fases. Outra desvantagem é a violação do "princípio da separação de responsabilidades", que diz que é melhor concentrar os esforços num aspecto de cada vez, i.e., os detalhes funcionais e técnicos devem ser considerados em fases separadas.

Soluções do capítulo 7

Exercício 7.1

Apenas fornecemos solução para a questão 7.1(b). A figura S7.1 mostra o modelo do processo da situação actual. Mais uma vez, decidimos não modelar os disparos: na realidade a maioria das tarefas requer um disparo.

Tarefas

1. Registar cliente privado
2. Registar cliente de negócios
3. Verificar licença
4. Dar licença em branco
5. Devolver licença incorrecta
6. Receber licença preenchida
7. Arquivar licença correcta
8. Verificar licença correcta
9. Iniciar viagem de negócios
10. Enviar cópia para ao DF (Departamento de Finanças)
11. Iniciar viagem privada
12. Verificação aceite
13. Preparar proposta
14. Preparar nova proposta
15. Chamar o cliente para aprovação
16. Enviar um memorando positivo
17. Verificar a proposta aprovada
18. Verificar a viagem privada
19. Determinar os custos dos voos
20. Contactar o cliente
21. Enviar um memorando negativo
22. Efectuar os pagamentos adiantados
23. Preparar o dinheiro e os cheques
24. Pagar taxas de registo
25. AND-split
26. Verificar a decisão
27. Verificar os pagamentos dos voos
28. AND-split
29. AND-split
30. Marcar reunião
31. O cliente paga
32. Enviar ao DF os custos dos voos de negócio
33. Pagar pelos voos
34. Verificar a viagem privada paga
35. Verificar a viagem de negócios paga
36. Verificar todos os pagamentos
37. Reservar o hotel

38. Enviar o dinheiro e os cheques
39. Imprimir o voucher
40. Verificar todas as reservas
41. Imprimir os bilhetes
42. Nada devolvido
43. Preparar uma pasta com os documentos
44. O cliente devolve o dinheiro e os cheques
45. Transfer
46. Fim da viagem privada
47. Receber a declaração
48. Processar a declaração
49. Calcular o saldo
50. Deduzir do orçamento
51. AND-split
52. Verificar a aprovação
53. Correcto
54. Fim da viagem de negócios
55. Estabelecer o saldo
56. Fechar o ficheiro
57. Enviar a notificação
58. Receber a licença não preenchida

Condições
c1. Viagem privada registada
c2. Viagem de negócios registada
c3. Sem licença
c4. O cliente preenche a licença
c5. Preencher a licença
c6. Licença correcta
c7. Licença incorrecta
c8. Licença arquivada
c9. Cópia enviada
c10. Cópia do ficheiro enviado
c11. Iniciar organização da viagem
c12. Não permitida
c13. Permitida
c14. Organizar a viagem privada
c15. Proposta
c16. Memorando positivo
c17. Pagamento adiantado
c18. Cliente aprovado/reprovado
c19. Taxa de registo
c20. Sem horário
c21. Horário
c22. Pedido de dinheiro e cheques
c23. Memorando negativo
c24. Cliente decidido
c25. Conhecidos os custos dos voos
c26. Dinheiro e cheques
c27. Pagar viagem privada

c28. Pagar viagem privada e voos privados a serem pagos
c29. Alguns voos privados
c30. Todos os voos de negócios
c31. Voos privados a serem pagos
c32. Voos de negócios a serem pagos
c33. Taxa calculada
c34. Cliente a pagar
c35. Pagar viagem de negócios
c36. Informação detalhada do preço
c37. Pagamentos completos
c38. Viagem privada paga
c39. Cliente pagou
c40. Voos privados pagos
c41. Voos de negócios pagos
c42. Esperar pela declaração
c43. Hotéis reservados
c44. Reservar hotéis
c45. Tem bilhetes impressos
c46. Dinheiro e cheques enviados
c47. Voucher impresso
c48. Esperar retorno
c49. Transporte tratado
c50. Bilhetes impressos
c51. Informação sobre a quantia
c52. Pasta de documentos pronta
c53. Transferes concluídos
c54. Declaração recebida
c55. Saldo
c56. Processar declaração
c57. Deduzido
c58. Quantia a deduzir
c59. Saldo aprovado
c60. Saldo reprovado
c61. Ficheiro a fechar
c62. Saldo estabelecido
c63. Saldo aprovado
c64. Incapaz de garantir a viagem

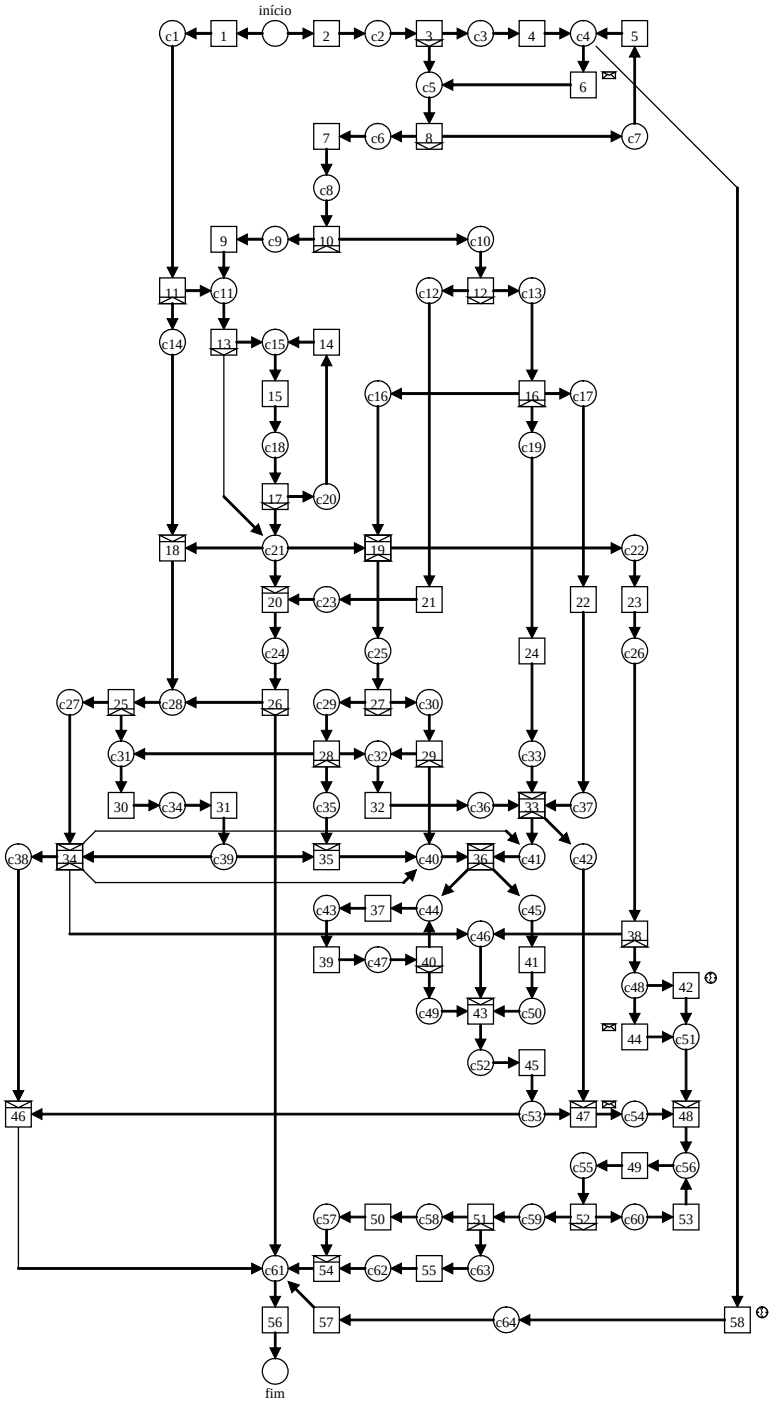


Figura S7.1. O processo da agência de viagens da Universidade Algures

GLOSSÁRIO

ActionWorkflow – O ActionWorkflow é um sistema de gestão de *workflow* que se concentra na coordenação de pessoas.

Actividade – Uma actividade é a execução de uma tarefa. Em contraste com uma tarefa, uma actividade está relacionada a um caso específico.
Sinónimos
- instancia de uma tarefa (*task instance*);
- disparo de uma transição (*transition firing*);
- operação (*operation*); e
- activity.

Actor – Um actor é uma pessoa, uma máquina ou uma unidade organizacional que está directamente ou indirectamente envolvida na execução de trabalho. Um actor desempenha o papel de um empreiteiro e/ou subempreiteiro.
Sinónimos
- jogador;
- actor; e
- player.

AND-join – Um AND-join é uma tarefa que só pode ser executada quando certas condições estão satisfeitas. Podemos comparar um AND-join com uma fase de uma cadeia de montagem que só pode ser executada quando todos os componentes necessários estão disponíveis. Um AND-join só é aplicado no momento em que vários workflows paralelos necessitam de ser sincronizados. Usando um AND-join é possível coordenar vários workflows paralelos para um caso particular.
Sinónimos
- junção (*join*);
- encontro (*rendezvous*); e
- tarefa de sincronização (*synchronization task*).

AND-split – Uma tarefa AND-split é o oposto lógico de uma tarefa AND-join. A execução de um AND-split resulta na criação de mais do que um *workflow* paralelo para o mesmo caso. Podemos afirmar que um AND-split divide um caso em várias partes as quais podem ser trabalhadas em simultâneo.
Sinónimos
- split; e
- fork.

API – O acrónimo API significa *Application Programing Interface*. A maioria dos sistemas de gestão de workflows oferece APIs para a sua integração com outras aplicações. No contexto da gestão de workflows, podemos também usar o termo WAPI (*Workflow Application Programing Interface*) em vez de API.

Aplicação – O sistema de gestão de *workflow* só controla os aspectos logísticos de um caso. O seu conteúdo é normalmente suportado por outras ferramentas, tais como processadores de texto ou folhas de cálculo. Chamamos a estas ferramentas aplicações. O desempenho de uma tarefa para um caso particular pode levar à iniciação de uma aplicação. Nestes casos, aplicações separadas podem ser integradas pelo sistema de gestão de *workflow* para formar um todo.
Sinónimos

- Programa externo; e
- Ferramenta.

Arquitectura – A arquitectura de um sistema (de *workflow*) é a sua estrutura na forma de componentes e na forma como interagem entre si (*interfacing*). A estrutura é normalmente hierárquica com uma distinção entre as infra-estruturas funcionais e as técnicas. A arquitectura funcional é baseada na estrutura dos componentes lógicos do sistema. A arquitectura técnica refere-se essencialmente aos componentes de *hardware* e de *software*.

Atribuição – Uma atribuição é descrita numa especificação que descreve claramente quais as tarefas a serem executadas para completar um caso específico, e em que ordem e intervalo de tempo têm de ser executadas.
Sinónimos

- assignment; e
- ordem.

Atributo de um caso – A forma como um caso progride através de um *workflow* depende das suas características inerentes e específicas. Vários atributos conseguem ser identificados para cada caso. Uma actividade pode alterar o valor desses atributos. Naturalmente, um caso utiliza só os seus atributos. Estes atributos são usados para encaminhar um caso. Por exemplo, a decisão resultante de um OR-split pode ser baseada nos atributos associados a um caso.
Sinónimos

- parâmetro operacional;
- variável do caso; e
- case attribute.

Audit Trail – O *autit trail* é um arquivo electrónico no qual o histórico de um *workflow* é registado. Contem vários detalhes sobre cada caso, tal como quando começou, o desempenho das tarefas executadas e os recursos reservados.
Sinónimos

- log file;
- ficheiro *log*; e
- registo.

Caso – Um caso é o que um sistema de gestão de *workflow* controla. Podemos também defini-lo como um "produto em progresso". Exemplos de casos podem incluir um pedido de seguro, a entrega de uma declaração de IRS, uma encomenda ou um plano de tratamento num hospital. Cada caso tem uma identidade única. Adicionalmente, um caso está sempre num estado particular de desenvolvimento em qualquer instante.
Sinónimos

- projecto;
- negócio;
- produto;
- serviço;
- ciclo de processo;
- instância de *workflow*; e
- case.

Casos de uso – Um caso de uso é um caso de um processo de *workflow* que é usado para descrever, demonstrar, especificar ou testar um processo ou sistema. O conjunto dos casos de uso devem cobrir os casos mais característicos e representativos, incluindo erros e excepções.
Sinónimos

- caso de negócio; e
- cenário;

Classe de recursos – Os recursos só podem executar um número limitado de tarefas. Para tornar fácil indicar – quando definimos um processo – qual o recurso que pode executar uma determinada tarefa, estes são agrupados em classes de recursos. Um recurso poderá pertencer a várias classes. O agrupamento dos recursos é geralmente feito de duas formas. Primeiro, os recursos são divididos com base na posição que ocupam na organização. Isso resulta em classes de recursos que são também conhecidas por unidades organizacionais: por exemplo, "Departamento de compras," "Grupo A," ou "Sucursal de Lisboa." Segundo, podem ser divididos por características funcionais – também conhecidas por papéis. Exemplos de papéis incluem "Executivo C," "Analista de Informação," e "Programador de COBOL." Cada um destes papéis corresponde a uma classe de recursos. As categorias que não são baseadas em papéis ou em unidades organizacionais são denominadas de classes de recursos livres.
Sinónimos

- categoria de recurso;
- grupo; e
- tipo de recurso;

Classificação de recurso – Os recursos – tanto os funcionários como os dispositivos automatizados – só podem executar um número limitado de tarefas. O que estas são depende de factores tais como quais as tarefas que um recurso pode executar e a localização onde tem de ser realizada. A classificação de recursos divide-os em subconjuntos, também conhecidos por classes de recursos. Exemplos de classificação de recursos incluem a separação em papéis ou em unidades organizacionais. Recursos com as mesmas características num

sistema de classificação formam uma classe específica de recursos. Alguns sistemas de gestão de *workflow* permitem que os relacionamentos entre as classes de recursos possam ser ilustradps esquematicamente.
Sinónimos

- diagrama organizacional;
- esquema de organização; e
- modelo de papéis.

Computer-Supported Cooperative Work (CSCW) – O trabalho cooperativo suportado por computador é o nome colectivo para os métodos, as técnicas e os sistemas que suportam a realização cooperativa de trabalho. Os produtos de *groupware* e sistemas de gestão de *workflow* são exemplos de CSCW.

Condição – Antes que uma tarefa possa ser executada no contexto de um caso particular, esse caso tem que preencher determinadas condições. A condição é um requerimento necessário antes que uma actividade possa ocorrer. Quando todas as condições para uma tarefa num caso particular estão satisfeitas, essa tarefa pode ser executada.
Sinónimos

- lugar; e
- place.

Contrato – Um contracto é um acordo vinculativo entre o empreiteiro e o subempreiteiro.

Correcto – Correcto é um critério de correcção definido para as redes de *workflow*, isto é, redes de Petri que representam processos de *workflow*. Uma rede de *workflow* é dita correcta se, para qualquer caso gerado pelo *workflow*, o processo ou *workflow* eventualmente termina e no momento em que o processo termina existe um só testemunho no lugar final e todos os outros lugares estão vazios. Mais, não deverão existir transições mortas; por outras palavras, deverá sempre ser possível executar uma tarefa arbitrária seguindo um caminho apropriado através da rede do *workflow*.
Sinónimo

- *sound*.

COSA – O COSA é um sistema de gestão de *workflow* baseado em redes de Petri da Software Ley (consultar http://www.cosa.de)

Dados de aplicação – São os dados usados pelos programas externos, em vez de serem geridos pelo sistema de *workflow*. O sistema de *workflow* não consegue aceder a estes dados directamente. Podem, no entanto, ser acedidos indirectamente pelos atributos de um caso e pelas próprias aplicações.

Definição de um workflow – Uma definição de um *workflow* consiste na definição de um processo, um sumário dos recursos necessários e a classificação desses recursos em classes.

Disparo – Um item de trabalho só pode ser executado quando o estado do caso o permitir. Mas a execução actual de uma tarefa, normalmente, requer mais. Se o item de trabalho é para ser executado por uma pessoa, ela primeiro tem de "seleccioná-lo" da lista de itens de trabalho antes que esta possa passar a ser uma actividade. Por outras palavras, o item de trabalho só é executado quando um recurso toma a iniciativa. Nestes casos, referimo-nos a *triggering*: o item de trabalho é disparado (*triggered*) pelo recurso. Outras formas de disparos também são possíveis, por exemplo, através de um evento externo (a chegada de uma mensagem EDI) ou com base num momento específico (a geração de uma lista de compras às seis horas). Há três tipos de disparos: iniciado pelo recurso, gerado exteriormente e com base no tempo. Os itens de trabalho que têm de ser executados imediatamente – sem a intervenção de um recurso – não necessitam de um disparo.
Sinónimos
- activação;
- alerta; e
- *trigger*.

Empreiteiro – Um (sub)empreiteiro é um "recurso" que é responsável por um processo e que executa as actividades ordenadas pelo supervisor. Note-se que um empreiteiro pode portar-se como um supervisor ao sub contratar outros recursos.
Sinónimos
- sub-empreiteiro;
- dono do processo; e
- *contractor*.

Encaminhamento – A definição de um processo determina de que forma os casos são enaminhados pelas várias tarefas. Existem 4 tipos distintos de encaminhamento: sequencial, selectivo, paralelo e iterativo.

Encaminhamento paralelo – Duas ou mais tarefas relacionadas a um caso específico podem ser executas em paralelo se, por definição, o processo contiver um AND-split e um AND-join. O AND-split permite que mais do que uma tarefa possa ser iniciada ao mesmo tempo. Ao ser completada a tarefa, os workflows paralelos são resincronizados usando um AND-join.

Encaminhamento selectivo – Porque a maior parte dos processos necessita de tratar de vários tipos de casos, nem todos os casos prosseguem num determinado processo da mesma forma, i.e. seguindo o mesmo caminho. Por outras palavras, podem existir vários caminhos num processo. Para assegurar que – dependendo das características de um caso – um caminho particular seja tomado, podemos usar um OR-split ou um OR-join. Para cada caso em execução, um OR-split permite seleccionar um conjunto de tarefas alternativas que irão ser executadas no contexto de um caso específico. Estes caminhos diferentes podem posteriormente convergir usando um OR-join.
Sinónimos
- encaminhamento alternativo;
- encaminhamento condicional; e

- selecção.

Encaminhamento sequencial – Referirmo-nos ao encaminhamento sequencial quando várias tarefas são executadas em sequência. Quando duas tarefas sucessivas estão ligadas por uma transição, então têm que ser executadas sequencialmente.
Sinónimos
- sequenciação; e
- sucessão.

Estado do caso – Em qualquer instante, um caso tem um estado particular que é determinado pelas condições que foram atingidas e pelos valores dos atributos associados ao caso.

Estado do workflow – O estado de um *workflow* é a "soma" dos estados dos casos, dos estados dos recursos e dos disparos.

Estrutura organizacional – Uma estrutura organizacional é uma estrutura em árvore que, graficamente, ilustra relacionamentos de autoridade. Por outras palavras, mostra a estrutura hierárquica das posições dentro da organização.

ExSpect – O ExSpect é uma ferramenta de simulação baseada em redes de Petri (ver http://www.exspect.com).

Factor crítico de sucesso – Um factor crítico de sucesso é um parâmetro (verbalmente expresso) de um processo ou sistema que é essencial no desempenho desse sistema ou processo.

Ferramenta de definição de workflows – É uma ferramenta usada para definir processos e para classificar recursos.
Sinónimo
- modelador de *workflow*; e
- workflow modeler;

Gestão de conhecimento – A gestão do conhecimento é o processo de colecção, enriquecimento e distribuição de conhecimento. O objectivo da gestão do conhecimento é certificar que o conhecimento certo, esteja no momento certo, disponível à pessoa que necessita desse conhecimento para cumprir uma tarefa.

Gestão de recursos – Para cada caso, um número de tarefas têm de ser executadas. Estas são executadas pelos recursos. Porque o número de recursos é limitado, é necessário harmonizar as actividades que necessitam de ser executadas com a capacidade dos recursos disponíveis. Isto é o que chamamos de gestão de recursos.
Sinónimos
- associação;
- reserva;
- alocação; e

- gestão da carga de trabalho.

Gestão de workflows – O termo gestão de *workflow* refere-se às ideias, métodos, técnicas e software usados para suportar processos de negócio estruturados. O objectivo da gestão de workflows é atingir processos de trabalho eficientes e fáceis de manter.
Sinónimos
- suporte de workflows; e
- WFM.

Gestor da lista de trabalho – Um sistema de gestão de *workflow* assegura que os itens de trabalho são atribuídos aos recursos. Se um item de trabalho é enviado para um funcionário, este aparecerá na sua lista de trabalho. Esta contém sempre uma lista das tarefas ainda por executar. Um funcionário, ao seleccionar um item de trabalho da lista de trabalho, pode imediatamente começar a executar essa tarefa. Note-se que um item de trabalho pode aparecer em mais do que uma lista.
Sinónimos
- lista da tarefas a executar;
- lista de entrada;
- *work tray*;
- *in-tray*;
- *worklist handler*;
- *worklist*; e
- *to-do-list*;

Gestor de casos – Um gestor de casos é a pessoa que é responsável por todos os casos ou um conjunto de várias tarefas de um caso.

Gestor de processo – Um gestor de processo é responsável pelo processo: a terminação dos casos e a reserva de recursos.
Sinónimo
- supervisor do processo.

Groupware – O termo *groupware* é um nome colectivo de produtos de *software* que permite a cooperação entre grupos de trabalho. O termo *groupware* está relacionado com o termo CSCW. O software de gestão de *groupware* e de *workflows* são frequentemente usados em combinação. Produtos típicos de groupware centram-se essencialmente na cooperação entre pessoas, enquanto que a ênfase dos sistemas de *workflow* é no suporte aos processos de negócios.

InConcert – O InConcert é um dos poucos sistemas de gestão de *workflow* ad hoc. Cada caso tem uma definição privada do processo que permite alterações dinâmicas e o desenho de *workflows* por descoberta.

Indicador de desempenho – Um indicador de desempenho é uma quantidade que é usada para medir um factor crítico de sucesso de um processo ou sistema. Exemplos de indicadores de desempenho são as médias do *flow time* (fluxo de tempo), utilização e nível de serviço.

Interoperabilidade – O termo interoperabilidade refere-se à habilidade de aplicações separadas poderem comunicar e cooperar entre si. Porque um sistema de *workflow* integra e interliga aplicações diferentes, o termo interoperabilidade certamente aplica-se a ele. A interoperabilidade mútua entre sistemas de *workflow* também é crucial para o sucesso da gestão de *workflows* em grandes organizações.

Interoperabilidade de workflows – A interoperabilidade entre workflows é o grau associado à capacidade de dois ou mais motores de workflows conseguirem trabalhar em conjunto enquanto executam um *workflow* comum partilhado. Isto abrange, por exemplo, a troca de casos e a contratação de itens de trabalho.

Item de trabalho – Um item de trabalho é a combinação de um caso e de uma tarefa que está prestes a ser executada. Tal como uma actividade um item de trabalho está associado a um caso específico. O item de trabalho desaparece imediatamente no momento em que começa a ser executado – no momento em que a tarefa se executa. Então, transforma-se numa actividade. Note-se que é possível, com base no estado do caso, determinar quais os itens de trabalho que estão à espera de ser executados.
Sinónimos
- atribuição de trabalho; e
- *work item*;

Iteração – A iteração é possível num processo de *workflow* se a sua estrutura permitir que uma ou mais tarefas sejam executadas repetidamente. Uma iteração pode, por exemplo, resultar de um controlo de qualidade: se o resultado da tarefa é insatisfatório, tem de ser repetida.
Sinónimos
- *workflow loop*;
- repetição; e
- ciclo.

JAD – O *Joint Application Design* (JAD) é uma abordagem de desenvolvimento de especificações durante um processo RAD usando *workshops* interactivos.

Lugar – Os lugares são os componentes passivos de uma rede de Petri. Um lugar pode conter nenhum, um, ou mais testemunhos. No processo de modelação de workflows, as condições são sempre representadas por lugares.
Sinónimos
- condição;
- canal; e
- *place*.

Método IPSD – IPSD é um acrónimo que designa o desenvolvimento de sistemas iterativos orientados a processos. O método IPSD combina elementos RAD e BPR para produzir uma abordagem de desenvolvimento de sistemas de *workflow*.

Modelo de referência – O modelo de referência do WFMC é uma definição arquitectónica na qual os seguintes componentes se distinguem: (1) Workflow Enactment Service; (2) Process Definition Tools; (3) Workflow Client Applications; (4) Invoked Applications; e (5) Administration and Monitoring Tools.

Motor de workflow – Um motor de *workflow* é responsável pela gestão dos workflows. Entre outras coisas, é responsável pela atribuição das tarefas, distribuição dos recursos, desempenho das actividades, preparação e modificação dos casos, lançamento das aplicações e a gravação de informação logística.
Sinónimos

- serviço de execução;
- *enactment service*; e
- *run-time engine*.

Organização em matriz – Uma organização em matriz está estruturada funcionalmente e hierarquicamente. A estrutura funcional é baseada em projectos de natureza temporária.

Organização em rede – Uma organização em rede consiste num conjunto de actores independentes que produzem bens e fornecem serviços. Porque não existe nenhuma relação de autoridade entre os actores, também denominamos esta estrutura por "empresa virtual."

Organização hierárquica – Numa organização hierárquica, as relações de autoridade têm uma estrutura em "árvore" que é frequentemente representada num organigrama.

OR-join – O OR-join é uma tarefa na qual um número de workflows alternativos reconverge. Contudo, ao contrário do AND-join nenhuma sincronização ocorre. Por outras palavras, a tarefa pode ser executada o mais rapidamente possível desde que uma única condição tenha sido satisfeita.
Sinónimo

- junção assíncrona.

OR-split – O OR-split é uma tarefa na qual uma escolha é efectuada. Durante o desempenho de um OR-split, um *workflow* é seleccionado através de um número de opções disponíveis. A escolha é normalmente baseada nos atributos particulares do caso em mão. No entanto, também pode ser aleatório. O OR-split é o oposto lógico do OR-join: um OR-split pode dividir um *workflow* num número de vias alternativas que mais tarde convergem num OR-join. Há dois tipos de tarefas OR-split: implícito e explícito. A diferença entre os dois tipos é baseada no momento em que a escolha é feita.
Sinónimos

- selecção;
- *switch*;
- escolha condicional; e

- ponto de decisão;

Papel – Para a execução de tarefas é indispensável possuir ou ter acesso a um conjunto de competências. Cada recurso – por exemplo uma pessoa – tem certas competências. Um papel define uma colecção de competências. Torna-se assim possível identificar o papel necessário para executar uma tarefa. É também indicado quais os papéis que cada recurso pode executar. Usando papéis, é possível assegurar que as tarefas serão atribuídas às pessoas certas. De facto, um papel é igual a uma classe de recursos baseada em características funcionais.
Sinónimos
- função;
- qualificação; e
- *role*.

Planeamento de capacidade – O planeamento de capacidade determina quantos recursos são reservados para determinada classe de recursos durante um período específico. Porque os casos estão muitas vezes dependentes de influências sazonais, padrões semanais e outras flutuações, a capacidade de planeamento concentra-se essencialmente em encontrar um equilíbrio entre os recursos necessários e os disponíveis.

Processo – A definição de um processo indica quais as tarefas que têm de ser executadas – e em que ordem – para completar um caso com sucesso. Por outras palavras, todos os caminhos possíveis são definidos. Um processo consiste em tarefas, condições e sub-processos. Usando AND-splits, AND-joins, OR-splits, ou OR-joins, fluxos paralelos e alternativos podem ser definidos. Os sub-processos também são compostos por tarefas, condições e possivelmente sub-processos adicionais. O uso de sub-processos pode permitir uma estrutura hierárquica de um processo complexo.
Sinónimos
- rede de workflow;
- WF-net;
- *flow chart*;
- *workflow script*;
- procedimento; e
- diagrama de processo.

Processo de negócio – Um processo de negócio centra-se na produção de determinados produtos. Podem ser produtos físicos, tais como um avião ou uma ponte, ou produtos menos tangíveis, tais como um desenho técnico, funções de consultoria ou avaliações. Por outras palavras, um "produto" pode também ser um serviço.
Sinónimo
- processo de trabalho; e
- *business process*.

Processo primário – É um processo para tratar de casos orientados ao cliente (*customer-oriented*). O processo centra-se na entrega de produtos ou serviços aos clientes da empresa.

Sinónimo
- processo de produção.

Processo secundário – É um processo que suporta os processos primários e que fornece recursos.
Sinónimo
- processo de suporte.

Processos ternários – Os processos ternários são os processos de gestão que controlam os processos primários e os secundários.
Sinónimos
- processos de gestão;
- processos executivos;

Protos – O Protos é uma ferramenta de BPR que facilita a modelação e distribuição de modelos de *workflow* (ver http://www.pallas-athena.com/).

Protótipo – Um protótipo é um sistema de software cuja funcionalidade é semelhante à de um sistema que ainda tem de ser produzido. Um protótipo também pode ser comparado com um modelo à escala.

RAD – O RAD, ou desenvolvimento rápido de aplicações, é um método de desenvolvimento de um sistema. O método RAD é caracterizado por um processo cíclico de desenvolvimento cuja cooperação com os utilizadores é prioritária.
Sinónimo
- Rapid Application Development.

Recurso – Um recurso é um meio de produção ou um grupo de tais meios. Poderá incluir actores, tais como pessoas, máquinas, meios de transporte, aplicações, departamentos e unidades de negócio. Os recursos só podem efectuar determinadas tarefas e são agrupados numa ou mais classes de recursos. A inclusão de um recurso numa categoria particular fornece informação acerca da posição que possui dentro da organização ou fornece informação sobre uma qualidade particular que possui.
Sinónimos
- agente;
- participante;
- meios de produção;
- utilizador;
- empregado; e
- funcionário;

Rede de Petri – Uma rede de Petri é a descrição de um processo em termos de lugares, transições e arcos. A semântica – o significado preciso – é sempre formalmente definida.
Sinónimos
- P/T net;

- rede P/T; e
- Petri net.

Rede de workflow – Uma rede de *workflow* é uma rede de Petri que representa um processo de *workflow*. Tal rede de *workflow* tem um lugar de origem e um lugar final. Todos os nós (i.e., lugar/condição ou transição/tarefa) estão num caminho que liga o lugar de origem e o lugar final. A rede de *workflow* é correcta se, para todos os casos, o processo eventualmente termina e no momento em que o processo termina um testemunho está no lugar final e todos os outros lugares estão vazios. Adicionalmente, não devem existir transições mortas, deverá ser possível executar uma tarefa arbitrária seguindo um caminho apropriado através da rede de *workflow*.
Sinónimo
- WF-net.

Redes de Petri de alto nível – Uma redes de Petri de alto nível é uma rede de Petri extendida para incluir as noções de cor, de tempo e de hierarquia. Esta extensão permite que processos complexos sejam descritos de uma forma simples.

Reengenharia de processos de negócio – A reengenharia de processos de negócio é uma reconsideração e reestruturação radical dos processos de negócio de forma a atingir melhoramentos drásticos nos custos, na qualidade e nos serviços.
Sinónimos
- BPR;
- redesenho de processos de negócio;
- regeneração do negócio; e
- *business process re-engineering*.

Rollback – Uma falha pode ocorrer durante a execução de uma tarefa ou actividade. Quando o *workflow* identifica uma falha, um *rollback* toma lugar. Por outras palavras, o sistema de *workflow* volta a posicionar-se no estado anterior à ocorrência da falha. Uma vez rectificada a falha, a actividade é executada novamente. Quando a actividade é completada com sucesso, ocorre um *commit* que torna as mudanças realizadas definitivas.

Sagitta 2000 – O Sagitta 2000 é o nome de um "novo" sistema de declarações alfandegárias do governo Holandês no qual a gestão de *workflows* tem um papel importante.

Simulação – A simulação é a imitação (no computador) da execução de um processo. Desta forma, o processo a ser simulado pode ser analisado graficamente e analiticamente.

Sistema de gestão de workflow – Um sistema de gestão de *workflow* é um pacote de software utilizado na implementação de um sistema de *workflow*. O termo refere-se a um sistema universalmente aplicável, por outras palavras, um sistema de gestão de *workflow* não está definido para uma situação de negócio

específica. Ao configurar um destes sistemas, ele é transformado num novo sistema que suporta workflows específicos. Ao contrário de um sistema de *workflow*, um sistema de gestão de *workflow* é uma aplicação genérica.
Sinónimo

- WFMS.

Sistema de processamento de transacções – Um sistema de processamento de transacções é um sistema de informação que regista, transforma e comunica detalhes relevantes do fluxo dos estados de um sistema.

Sistema de workflow – Um sistema *workflow* é um sistema que suporta workflows numa situação específica de negócio. Ao contrário de um sistema de gestão de *workflow*, um sistema de *workflow* é adaptado para um aplicação particular. Um sistema de *workflow* consiste normalmente num sistema de gestão de *workflow*, mais os processos, a classificação de recursos, as aplicações, os sistemas de base de dados, etc. Podemos comparar a diferença entre um sistema de gestão de *workflow* e um sistema de *workflow* estabelecendo uma analogia com um sistema de gestão de base de dados e um sistema de base de dados.
Sinónimo

- WFS; e
- *workflow system*.

Staffware – O Staffware é um dos principais sistemas de gestão de *workflow* (ver http://www.staffware.com/)

Supervisor – Um supervisor é um actor que quer que uma actividade seja executada pelo fornecedor, o supervisor contrata trabalho a um subfornecedor. Nos termos de um desses contractos, o supervisor e o fornecedor estabelecem acordos sobre a natureza do trabalho, planeamento, e os custos envolvidos. Dentro do contexto da organização interna, o termo supervisor também inclui um patrão ("boss").
Sinónimos

- cliente;
- dono do caso; e
- dono do fluxo.

Tarefa – Uma tarefa é um processo "atómico": um que não pode ser subdividido em componentes. Por isso é uma unidade lógica de trabalho, por outras palavras, uma tarefa ou é totalmente executada ou não é executada. Uma tarefa não está ligada a um caso específico. Quando uma tarefa é executada para um caso específico, designamo-la de actividade. Também diferenciamos tarefas manuais de automáticas e semi-automáticas. Uma tarefa manual é executada por uma pessoa, sem qualquer intervenção de uma aplicação (por exemplo, a assinatura manuscrita de um documento). Uma tarefa automática é executada por uma aplicação sem qualquer intervenção humana. Uma tarefa semi-automática envolve uma aplicação interactiva (por exemplo, um processador de texto).
Sinónimos

- tarefa de um processo;

- passo de um processo;
- passo de trabalho; e
- transição;

Testemunho – O estado de uma rede de Petri é determinado pela distribuição dos testemunhos pelos lugares da rede. Se os workflows são transformados em redes de Petri, o estado de um caso corresponde à localização de um ou mais testemunhos.
Sinónimos
- objecto; e
- *token*.

Tipo de casos – Casos similares pertencem ao mesmo tipo de caso. Há uma correspondência de um para um entre os tipos de caso e os processos. Por outras palavras, uma definição de um processo pertence a um tipo de caso específico.

Transacção – Uma transacção é um protocolo de troca que resulta na emissão de um contrato para uma actividade.

Transição – As transições são os componentes activos das redes de Petri. O disparo de uma transição resulta na mudança do estado da rede. Na modelação de workflows, uma transição normalmente coincide com uma tarefa.
Sinónimos
- evento; e
- processador.

Triagem – A triagem é a selecção e o estabelecimento da prioridade dos casos durante a execução de uma tarefa, com base em características fáceis de identificar (um exemplo, é a linha expresso de um supermercado, onde os casos estão divididos em casos grandes – que requerem muito trabalho – e casos pequenos – que exigem menos trabalho). O objectivo da triagem é reduzir o tempo médio de execução das tarefas.

UML – O UML (*Unified Modeling Language*) é um *standard* para o desenvolvimento de software. É uma linguagem gráfica para a visualização, especificação, construção e documentação de artefactos de um sistema de software. Contudo, o uso do UML não é restrito ao desenvolvimento de software. Alguns dos seus diagramas são também usados para modelar empresas, negócios, análise de processos e configuração de sistemas.

Unidade organizacional – Uma unidade organizacional é uma equipa de funcionários que geralmente trabalha em grupos. A composição de determinado grupo é baseada na localização do trabalho, de acordo com os papéis comuns a serem desempenhados, ou um conjunto de tarefas. Tais situações referem-se, respectivamente, a uma estrutura de grupo geográfica, funcional ou baseado em processos. Um grupo de pessoas que trabalhem em cooperação, sob a liderança de um gestor, nas suas próprias tarefas e com as suas próprias responsabilidades é chamado de unidade organizacional. Uma organização é frequentemente

dividida em unidades organizacionais seguindo uma estrutura hierárquica, sendo possível que uma unidade possa fazer parte de outra unidade. Deverá ser possível identificar a unidade organizacional responsável para cada tarefa. Também é possível que isto dependa do próprio caso, por exemplo, as hipotecas de mais de $200.000, são tratadas pela unidade A. Cada recurso é "possuído" por uma unidade organizacional. De facto, tal unidade não é mais que a classe do recurso, baseada nas características organizacionais.
Sinónimos

- departamento; e
- equipa.

Woflan – O Woflan é um analisador de *workflows* com base em redes de Petri (consultar http://www.tm.rue.nl/it/woflan).

Workflow – Um *workflow* inclui casos, recursos e disparos que estão relacionados com um processo específico.

Workflow Ad hoc – Em geral, muitos casos diferentes envolvem o mesmo processo de negócio. Contudo, em certos casos é necessário modificar o processo para um caso específico. Designamos esse processo de *workflow* ad hoc.

Workflow Managment Coalition – É uma organização internacional constituída por utilizadores, fornecedores, fabricantes e programadores de produtos de *workflow*. O objectivo mais importante desta organização é desenvolver produtos *standards* no campo dos sistemas de *workflow*. Os resultados obtidos são publicados em canais tais como o World Wide Web (http://www.aiim.org/WfMC).
Sinónimo

- WFMC.

(Página deixada propositadamente em branco)

BIBLIOGRAFIA

Gestão de Workflow

W.M.P. van der Aalst and K.M. van Hee. *Workflow Management: Modellen, Methoden en Systemen (in Dutch)*. Academic Service, Schoonhoven, 1997.

D. Georgakopoulos, M. Hornick, and A. Sheth. An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure. *Distributed and Parallel Databases*, 3:119-153, 1995.

S. Jablonski and C. Bussler. *Workflow Management: Modeling Concepts, Architecture, and Implementation*. International Thomson Computer Press, London, UK, 1996.

T.M. Koulopoulos. *The Workflow Imperative*. Van Nostrand Reinhold, New York, 1995.

F. Leymann and D. Roller. *Production Workflow: Concepts and Techniques*. Prentice-Hall PTR, Upper Saddle River, New Jersey, USA, 2000.

T. Schäl. *Workflow Management for Process Organizations*, volume 1096 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1996.

A.P. Sheth, W.M.P. van der Aalst, and I.B. Arpinar. Processes Driving the Networked Economy: ProcessPortals, ProcessVortex, and Dynamically Trading Processes. (Special issue on Workflow Management Systems) *IEEE Concurrency*, 7(3):18-31, 1999.

Workflow Management Coalition

P. Lawrence, editor. *Workflow Handbook 1997, Workflow Management Coalition*. John Wiley and Sons, New York, 1997.

WFMC. Workflow Management Coalition Terminology and Glossary (WFMC-TC-1011). Technical report, Workflow Management Coalition, Brussels, 1996.

Workflow Management Coalition. WFMC Home Page. http://www.wfmc.org.

Gestão de Processos de Negócio/Re-engenharia

W.M.P. van der Aalst, J. Desel, and A. Oberweis, editors. *Business Process Management: Models, Techniques, and Empirical Studies*, volume 1806 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 2000.

W.M.P. van der Aalst. On the automatic generation of workflow processes based on product structures. *Computers in Industry*, 39:97-111, 1999.

W.M.P. van der Aalst and K.M. van Hee. Business Process Redesign: A Petri-net-based approach. *Computers in Industry*, 29(1-2):15-26, 1996.

T.H. Davenport. *Process innovation : re-engineering work through information technology*. Harvard Business School Press, Boston, 1993.

M. Hammer. Re-engineering work: Don't automate, Obliterate. *Harvard Business review*, pages 104-112, July/August 1990.

M. Hammer and J. Champy. *Re-engineering the corporation*. Nicolas Brealey Publishing, London, 1993.

T.W. Malone, W. Crowston, J. Lee, B. Pentland, and et. al. Tools for Inventing Organizations: Toward a Handbook for Organizational Processes. *Management Science*, 45(3):425-443, 1999.

D. Morris and J. Brandon. *Re-engineering your business*. McGraw-Hill, New York, 1993.

Rapid Application Development

J. Martin. *Rapid Application Development*. MacMillan, New York, 1991.

Redes de Petri

W.M.P. van der Aalst. Putting Petri nets to work in industry. *Computers in Industry*, 25(1):45-54, 1994.

J. Desel and J. Esparza. *Free Choice Petri Nets*, volume 40 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge, UK, 1995.

K.M. van Hee. *Information System Engineering: a Formal Approach*. Cambridge University Press, 1994.

K. Jensen. *Colored Petri Nets. Basic Concepts, Analysis Methods and Practical Use*. EATCS monographs on Theoretical Computer Science. Springer-Verlag, Berlin, 1996.

T. Murata. Petri Nets: Properties, Analysis and Applications. *Proceedings of the IEEE*, 77(4):541-580, April 1989.

J.L. Peterson. *Petri net theory and the modeling of systems*. Prentice-Hall, Englewood Cliffs, 1981.

W. Reisig. *Petri Nets: An Introduction*, volume 4 of *EATCS Monographs in Theoretical Computer Science*. Springer-Verlag, Berlin, 1985.

W. Reisig and G. Rozenberg, editors. *Lectures on Petri Nets I: Basic Models*, volume 1491 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1998.

W. Reisig and G. Rozenberg, editors. *Lectures on Petri Nets II: Applications*, volume 1492 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1998.

Modelar de Workflows com redes de Petri

W.M.P. van der Aalst. The Application of Petri Nets to Workflow Management. *The Journal of Circuits, Systems and Computers*, 8(1):21-66, 1998.

N.R. Adam, V. Atluri, and W. Huang. Modeling and Analysis of Workflows using Petri Nets. *Journal of Intelligent Information Systems*, 10(2):131-158, 1998.

C.A. Ellis. Information Control Nets: A Mathematical Model of Office Information Flow. In *Proceedings of the Conference on Simulation, Measurement and Modeling of Computer Systems*, pages 225-240, Boulder, Colorado, 1979. ACM Press.

C.A. Ellis and G.J. Nutt. Modeling and Enactment of Workflow Systems. In M. Ajmone Marsan, editor, *Application and Theory of Petri Nets 1993*, volume 691 of *Lecture Notes in Computer Science*, pages 1-16. Springer-Verlag, Berlin, 1993.

Sistemas de Gestão de Workflow e Ferramentas

W.M.P. van der Aalst, P. de Crom, R. Goverde, K.M. van Hee, W. Hofman, H. Reijers, and R.A. van der Toorn. ExSpect 6.4: An Executable Specification Tool for Hierarchical Colored Petri Nets. In M. Nielsen and D. Simpson, editors, *Application and Theory of Petri Nets 2000*, *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, 2000.

FileNET. *Ensemble User Guide*. FileNET Corporation, Costa Mesa, California, 1998.

J. Hernandez. The SAP R/3 Handbook, 1997.

InConcert. *InConcert Process Designer's Guide*. InConcert Inc., Cambridge, Massachusetts, 1997.

IBM. *IBM MQseries Workflow: Concepts and Architecture*. IBM Corporation, Armonk, USA, 1999.

Pallas Athena. *Protos User Manual*. Pallas Athena BV, Plasmolen, The Netherlands, 1999.

Y. Perreault and T. Vlasic. *Implementing Baan IV*. Macmillan Computer Publishing, New York, 1998.

Promatis. *Income Workflow User Manual*. Promatis GmbH, Karlsbad, Germany, 1998.

Software-Ley. *COSA User Manual*. Software-Ley GmbH, Pullheim, Germany, 1998.

Staffware. *Staffware 2000 / GWD User Manual*. Staffware plc, Berkshire, United Kingdom, 1999.

H.M.W. Verbeek and W.M.P. van der Aalst. Woflan 2.0: A Petri-net-based Workflow Diagnosis Tool. In M. Nielsen and D. Simpson, editors, *Application and Theory of Petri Nets 2000*, *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, 2000.

Análise de Workflows

W.M.P. van der Aalst. Formalization and Verification of Event-driven Process Chains. *Information and Software Technology*, 41(10):639-650, 1999.

W.M.P. van der Aalst. Woflan: A Petri-net-based Workflow Analyzer. *Systems Analysis - Modeling - Simulation*, 35(3):345-357, 1999.

W.M.P. van der Aalst, K.M. van Hee, and H.A. Reijers. Analysis of Discrete-time Stochastic Petri Nets. *Statistica Neerlandica*, 2000 (to appear).

W.M.P. van der Aalst and A.H.M. ter Hofstede. Verification of Workflow Task Structures: A Petri-net-based Approach. *Information Systems*, 25(1):43-69, 2000.

A.H.M. ter Hofstede, M.E. Orlowska, and J. Rajapakse. Verification Problems in Conceptual Workflow Specifications. *Data and Knowledge Engineering*, 24(3):239-256, 1998.

W. Sadiq and M.E. Orlowska. Applying Graph Reduction Techniques for Identifying Structural Conflicts in Process Models. In *Proceedings of the 11th International Conference on Advanced Information Systems Engineering (CAiSE '99)*, volume 1626 of *Lecture Notes in Computer Science*, pages 195-209. Springer-Verlag, Berlin, 1999.

Flexibilidade de Workflows

W.M.P. van der Aalst and T. Basten. Inheritance of Workflows: An approach to tackling problems related to change. *Theoretical Computer Science*, 2000 (to appear).

W.M.P. van der Aalst, T. Basten, H.M.W. Verbeek, P.A.C. Verkoulen, and M. Voorhoeve. Adaptive Workflow: On the Interplay between Flexibility and Support. In J. Filipe, editor, *Enterprise Information Systems*, pages 61-68. Kluwer Academic Publishers, Norwell, 2000.

C.A. Ellis, K. Keddara, and G. Rozenberg. Dynamic change within workflow systems. In N. Comstock, C. Ellis, R. Kling, J. Mylopoulos, and S. Kaplan, editors, *Proceedings of the Conference on Organizational Computing Systems*, pages 10 - 21, Milpitas, California, August 1995. ACM SIGOIS, ACM Press, New York.

P. Heinl, S. Horn, S. Jablonski, J. Neeb, K. Stein, and M. Teschke. A Comprehensive Approach to Flexibility in Workflow Management Systems. In *Work Activities Coordination and Collaboration (WACC'99)*, pages 79-88, San Francisco, February 1999. ACM press.

M. Reichert and P. Dadam. ADEPTflex: Supporting Dynamic Changes of Workflow without Loosing Control. *Journal of Intelligent Information Systems*, 10(2):93-129, 1998.

Workflows Inter-organizacionais

W.M.P. van der Aalst. Loosely Coupled Interorganizational Workflows: Modeling and Analyzing Workflows Crossing Organizational Boundaries. *Information and Management*, 37(2):67-75, March 2000.

W.M.P. van der Aalst. Process-oriented Architectures for Electronic Commerce and Interorganizational Workflow. *Information Systems*, 24(8):639-671, 2000.

W.M.P. van der Aalst. Interorganizational Workflows: An Approach based on Message Sequence Charts and Petri Nets. *Systems Analysis - Modeling - Simulation*, 34(3):335-367, 1999.

Investigação Operacional

K.R. Baker. *Introduction to Sequencing and Scheduling*. Wiley & Sons, 1974.

J.A. Buzacott. Commonalities in Re-engineered Business Processes: Models and Issues. *Management Science*, 42(5):768-782, 1996.

L. Kleinrock. *Queueing systems, Vol. 1:Theory*. Wiley-Interscience, London, 1975.

J.J. Moder and S.E. Elmaghraby. *Handbook of Operations Research: foundations and fundamentals*. Van Nostrand Reinhold,New York, 1978.

M. Pinedo. *Scheduling : theory, algorithms, and systems*. Prentice-Hall, Englewood Cliffs, 1995.

S.M. Ross. *A course in simulation*. Macmillan, New York, 1990.

Unified Modeling Language (UML)

G. Booch, J. Rumbaugh, and I. Jacobson, *The Unified Modeling Language User Guide,* Addison-Wesley, Reading, Massachusetts, 1998.

J. Rumbaugh, I. Jacobson, and G. Booch, *The Unified Modeling Language Reference Manual,* Addison-Wesley, Reading, Massachusetts, 1999.

C. Marshall, Enterprise Modeling With UML: Designing Successful Software Through Business Analysis, Addison-Wesley, Reading, Massachusetts, 2000.

(Página deixada propositadamente em branco)

Série

Ensino

•

Imprensa da Universidade de Coimbra

Coimbra University Press

2009