Poseidon: Uma aplicação de suporte ao desenho semi-automático de workflows

Jorge Cardoso

Departamento de Matemática e Engenharias
Universidade da Madeira
jcardoso@uma.pt

Resumo. Uma promissora solução de gestão, coordenação e orquestração de serviços electrónicos é o uso dos sistemas de gestão de workflows. Embora várias linguagens para modelar workflows, sistemas para a gestão de workflows e técnicas de análise formais tenham sido estudadas de um modo extensivo, a investigação de métodos para auxiliar o desenho de workflows é praticamente inexistente. O propósito deste trabalho é apresentar uma aplicação, denominada de Poseidon, para ajudar os analistas e designers de workflows no seu trabalho, permitindo a criação semi-automática de workflows com uma elevada qualidade. A aplicação Poseidon define um conjunto de passos que orientam o desenho de workflows com base em casos de negócio.

1 Introdução

A Web, o desenvolvimento do comércio electrónico (e-commerce), e novos conceitos arquitecturais tais como os serviços Web, criaram as bases para o aparecimento de uma nova economia interligada em rede (Sheth, Aalst et al. 1999). Com a maturidade das infra-estruturas que suportam o e-commerce, será possível às organizações incorporarem serviços Web como parte dos seus processos de negócio. Um vasto leque de modernos sistemas de gestão de workflows tem sido desenvolvido para suportar vários tipos de processos de negócio (Cardoso and Sheth 2003).

Os estudos e investigação realizados até a data na área da gestão de workflows centram-se em três domínios principais: arquitecturas de sistemas de workflow (Miller, Sheth et al. 1996), linguagem de especificação e modelação (Aalst and Hofstede 2003; BPML 2004; BPMN 2005; WS-BEPL 2005), e análise de workflows (Aalst 1998; Aalst 2000). Estas áreas de investigação são de reconhecida importância para a construção de robustos e sofisticados sistemas de gestão de workflows. Não obstante, uma área importante foi negligenciada, a investigação de metodologias para suportar o desenho de workflows. De facto, os estudos sobre o desenho de workflows são escassos. Em 1996, Sheth et al. (Sheth, Georgakopoulos et al. 1996) determinaram que a modelação e desenho de workflows e processos seria uma área de estudo importante que deveria ser investigada em mais profundidade. A utilização de metodologias adequadas para auxiliar o desenho de workflows é um elemento chave que

condiciona o sucesso de qualquer projecto de workflow e que requer a disponibilidade de aplicações específicas.

Devido a falta de aplicações apropriadas para assistir o desenho de workflows, desenvolvemos a aplicação Poseidon. A aplicação Poseidon é de grande utilidade tanto para os analistas de processos de negócio como para os desenhadores durante as duas primeiras fases do desenvolvimento do ciclo de vida dos workflows, i.e., a fase de identificação dos requisitos e a fase de desenho. Esta aplicação permite a criação de workflows de uma forma semi-automática. Embora existam várias ferramentas comerciais que permitem desenhar workflows, como por exemplo o ARIS Business Designer (ARIS 2005), TIBCO Staffware Process Suite (TIBCO 2002), COSA BPM (COSA 2005), MQSeries Workflow (IBM), METEOR designer (Kochut 1999), etc, estas aplicações não permitem o desenho semi-automático de workflows.

Este trabalho visa descrever a metodologia e funcionalidades da aplicação Poseidon. A aplicação pode ser usada durante a análise e a fase de desenho. Pode ser vista como uma metodologia que estrutura os vários passos que orientam o desenho de workflows baseada na compilação de requisitos obtidos da comunicação com as pessoas, gerentes e especialistas na área de modelação de processos de negócio.

Este trabalho está estruturado da seguinte forma. A secção 2 apresenta os requisitos e objectivos da aplicação Poseidon. Na secção 3 apresentamos e descrevemos a metodologia seguida pela aplicação Poseidon para suporta o desenho semi-automático de workflows. A secção 4 descreve alguns pormenores da implementação da aplicação. Finalmente, na secção 5 apresentamos as nossas conclusões.

2 Requisitos e objectivos da aplicação Poseidon

As equipas de desenvolvimento, os consultores e os académicos têm diferentes perspectivas acerca do desenvolvimento de processos de negócio e workflows. Algumas organizações vêem o desenvolvimento de workflows como uma actividade *ad-hoc* de desenho, outros vêem o desenvolvimento de workflows como um melhoramento ou o redesenho de processos de negócio isolados, e apenas uma minoria vêem-no como a reorganização de processos de forma compreensiva, utilizando metodologias e ferramentas de modelação para estruturar as actividades da organização num workflow bem definido.

Uma empresa pode escolher diferentes abordagens para desenhar diversos tipos de workflows. Não existe uma estrutura "certa" ou "errada" de desenvolvimento. Não obstante, algumas abordagens são mais apropriadas do que outras para o desenho de determinados tipos de workflows. Se uma abordagem inadequada for escolhida e usada então é muito provável que os workflows desenhados (Sheth, Aalst et al. 1999) tenham um valor e utilidade reduzidos. Tendo a noção que uma abordagem feita por medida pode não satisfazer as necessidades de diferentes empresas, desenvolvemos a aplicação Poseidon tendo por base os seguintes requisitos:

- **Simplicidade e facilidade de uso**: a aplicação e metodologia associada têm de ser facilmente entendidas pelos utilizadores finais;
- **Dimensão dos workflows**: a aplicação tem de suportar a modelação de workflows com uma dimensão pequena e mediana. Workflows pequenos são aqueles

que contêm aproximadamente 15 tarefas e workflows medianos são aqueles que têm aproximadamente 30 tarefas;

- **Estrutura dos workflows**: a aplicação tem de ser adequada para o desenho semi-automático de workflows de produção e de administração (McCready 1992), i.e., workflows mais estruturados, previsíveis e repetitivos;
- **Grau de automatização**: baseado numa das principais vantagens e objectivos dos sistemas de workflows, isto é a automatização, é requerido o desenvolvimento de uma aplicação e metodologia que permitam automatizar o maior número de etapas associadas ao desenho de workflows.

O ciclo de vida de um workflow é composto por várias fases, incluindo análise, desenho, implementação, testes e manutenção. Neste trabalho, estamos particularmente interessados na fase de desenho de workflows. O objectivo da aplicação Poseidon é fornecer uma metodologia implementada numa ferramenta que permita o desenho semi-automático de workflows. A aplicação permite auxiliar designers de workflows no seu trabalho e é independente da tecnologia utilizada na implementação ou programação do workflow. A aplicação fornece uma estrutura básica conceptual composta por passos, procedimentos e algoritmos que determinam como o desenho de workflows é empreendido.

3 Metodologia

Nesta secção iremos dar uma descrição geral sobre a metodologia seguida pela aplicação Poseidon para a construção de workflows. A metodologia envolve quatro fases principais e para cada uma apresentaremos mecanismos para o suporte e assistência a sua concretização.

A primeira fase (secção 3.1) da metodologia apoia-se fortemente na informação recolhida em entrevistas, sessões de grupos, “brainstorming” e reuniões entre as várias pessoas com conhecimento tácito sobre o funcionamento dos processos de negócio. Usaremos o termo ‘entrevistas’ para designar estes três métodos de obtenção de informação. As entrevistas deverão ser realizadas entre os analistas de workflows e as pessoas que conhecem bem a lógica dos processos de negócio. As pessoas que conhecem a lógica de negócio irão tipicamente incluir pessoal administrativo, gestores de departamentos, quadros gestores médios ou mesmo CEOs (Chief Executive Officer). Com base no conhecimento recolhido construímos uma tabela de casos de negócio (TCN). A propriedade básica de um processo é que este é baseado em casos. Isto significa que cada tarefa é executada com base num caso específico (Aalst 1998). A tabela captura os vários casos que um processo de negócios descreve.

Na segunda fase (secção 3.2) é extraído um conjunto de funções de escalonamento com base na informação presente na TCN. Como um workflow é composto por um conjunto de tarefas, para cada tarefa é extraída uma função de escalonamento. As funções são subsequentemente minimizadas. Uma função de escalonamento é uma função Booleana para a qual os parâmetros são as variáveis de negócios. Uma variável de negócio é uma variável do workflow presente nas condições que determinam o controlo do fluxo de tarefas. Tipicamente, essas variáveis estão associadas as transi-

ções que saem de das tarefas do tipo xor-split ou or-split (Cardoso 2005). Cada função modela o comportamento de uma tarefa ou tarefas em tempo de execução de um workflow, i.e., para um dado conjunto de variáveis de negócios e seus valores, a função de escalonamento indica se a tarefa ou conjunto de tarefas são executadas ou não em tempo de execução.

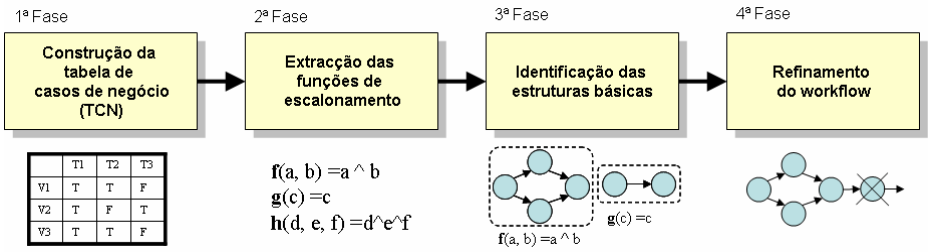


Fig. 1. Metodologia associada a aplicação Poseidon

A terceira fase (secção 3.3) consiste no uso das funções obtidas na fase anterior para identificar os blocos sequenciais e paralelos presentes num workflow. Um workflow pode ser decomposto em blocos básicos sequenciais, paralelos, e não determinísticos (Cardoso, Miller et al. 2004).

Na última fase (secção 3.4), integramos os blocos previamente identificados com os blocos não determinísticos existentes no workflow. O workflow é limpo de qualquer tarefa auxiliar introduzida sem valor para o workflow e, se necessário, o workflow pode ser ligeiramente reestruturado ou modificado por razões de claridade.

3.1 Tabela de casos de negócio

A propriedade básica de um processo é este ser baseado em casos de negócio (Aalst 1998). Isto significa que cada tarefa é executada no âmbito de um determinado caso de negócio. Os casos de negócio à muito que são adoptados no desenho de processos de negócio bem como no desenho de software. Para obtermos todos os casos de negócio representados num workflow, foi introduzido o conceito de tabela de casos de negócio (TCN). Esta tabela é uma ferramenta que permite capturar e descrever os casos de negócio de forma simples, mas poderosa.

Cada caso de negócio corresponde a uma entrada na TCN e estabelece as tarefas de workflow executadas com base no valor de variáveis de negócio. Variáveis de negócio são variáveis que influenciam o encaminhamento ou controlo do fluxo num workflow. As variáveis de negócio são identificadas durante a fase participativa em que são realizadas as entrevistas. Cada variável de negócio tem um domínio, também identificado durante a fase participativa. O domínio identifica os valores que a variável de negócio pode tomar. Por exemplo, num workflow que suporte os processos de negócio de um banco, a variável de negócio Valor de Empréstimo determina se o pedido de empréstimo será aceite ou rejeitado. Se a variável tiver um valor superior a 500 000 € então o empréstimo é rejeitado e a tarefa *'rejeitar'* é executada, caso contrário a tarefa *'aceitar'* é executada.

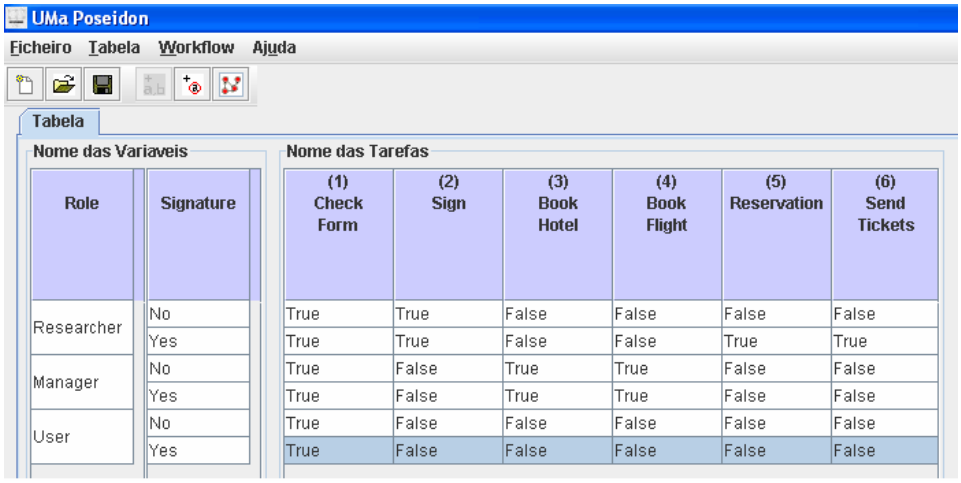
3.1.1 Esquema da tabela de casos de negócio

Uma tabela de casos de negócio é baseada numa tabela de duas dimensões. A estrutura da tabela é a seguinte. As colunas são divididas em duas classes. A primeira classe agrupa um conjunto de variáveis de negócio (vn_j), enquanto a segunda classe é constituída pelas tarefas (t_i) que fazem parte do workflow em questão. Cada célula na tabela relaciona o valor das variáveis de negócio com as tarefas. Em cada uma destas células é guardada a informação sobre se a tarefa será executada ou não (utilizamos os valores *true* e *false* para indicar se uma tarefa é executada ou não).

As primeiras células de cada linha, que pertencem às colunas da primeira classe, contêm os valores que podem ser atribuídos as variáveis de negócio. As células correspondentes às colunas da segunda classe contêm informação que indica se uma tarefa em particular deverá ser executada ou não em runtime. A ideia é estabelecer se uma dada tarefa será executada em runtime com base nos valores referentes a um determinado conjunto de variáveis. De maneira mais formal, o que pretendemos é obter para cada tarefa t_i, a seguinte função onde vn_i é uma variável de negócio:

$$\text{escalonamento}\ (t_i, vn_{1,}\, vn_{2,} \ldots, vn_j) \in \{\text{True, False}\} \qquad \textbf{(1)}$$

Uma célula pertencente às colunas da segunda classe poderá conter o símbolo de execução (i.e., True) ou o símbolo de não execução (i.e., False). A Figura 2 ilustra uma TCN preenchida.



Role	Signature	(1) Check Form	(2) Sign	(3) Book Hotel	(4) Book Flight	(5) Reservation	(6) Send Tickets
Researcher	No	True	True	False	False	False	False
	Yes	True	True	False	False	True	True
Manager	No	True	False	True	True	False	False
	Yes	True	False	True	True	False	False
User	No	True	False	False	False	False	False
	Yes	True	False	False	False	False	False

Fig. 2. Exemplo de uma TCN preenchida

3.1.2 Construção da tabela de casos de negócio

Perceber o esquema de uma TCN é relativamente simples, no entanto a sua construção é um pouco mais complexa. A metodologia (descrita em (Cardoso 2005)) para construir e preencher a tabela com casos de negócio é um processo iterativo e interactivo, baseado nas entrevistas realizadas.

No primeiro passo, é necessário aferir se as tarefas presentes na tabela são sempre executadas com base nos valores das variáveis de negócio. Se a tarefa é sempre exe-

cutada, esta recebe o símbolo *true*; se nunca é executada recebe o símbolo *false* (ver a figura 2). Caso contrário existe um conflito de símbolos. Os conflitos de símbolos indicam que a realização de um conjunto de tarefas depende de uma ou mais variáveis de negócio. O conflito verifica-se quando os dois símbolos possíveis (i.e., *true* e *false*) foram atribuídos a uma mesma célula da tabela de casos de negócio. Para resolver este conflito, o analista deverá identificar pelo menos uma variável de negócio que controla a realização da tarefa que se encontra em conflito. Depois dessa variável ter sido identificada é necessário realizar os seguintes passos:

1. Deverá ser adicionada uma coluna ao lado esquerdo da tabela de casos de negócio e deverão ser adicionadas as respectivas linhas para a coluna inserida.
2. A coluna deverá ter o nome da variável de negócio identificada.
3. Cada linha da tabela deverá ser duplicada *n-1* vezes, onde *n* é o domínio de valores possíveis para a nova variável de negócio introduzida.
4. Para cada célula, correspondente à nova coluna inserida, são atribuídos os valores do domínio da variável de negócio.

Depois da estrutura da tabela ser actualizada com a nova variável, as células referentes às tarefas deverão também ser actualizadas com os símbolos apropriados. Deverão ser realizadas novas entrevistas para que sejam novamente analisadas quais as tarefas que deverão ser realizadas em runtime com base nos valores referentes às variáveis de negócio presentes na tabela.

3.2 Extracção de funções de escalonamento

Na segunda fase, extraímos um conjunto de funções de escalonamento da TCN. Uma função de escalonamento é uma função lógica na qual os parâmetros são variáveis de negócio da TCN. Cada função modela a execução de tarefas em runtime, i.e. para um conjunto de variáveis de negócio, e respectivas asserções, a função indica se uma tarefa ou tarefas estão agendadas para execução ou não. Para extrair um conjunto de funções de escalonamento, é necessário em primeiro lugar mapear a TCN para uma tabela de verdade (truth table). O mapeamento poderá ser alcançado da seguinte forma:

- Para cada variável de negócio, determinar o número mínimo de bits *nmb* necessários para representar a variável. Representar cada bit com uma variável binária distinta (por exemplo, ‘a’, ‘b’, ‘c’, ...). Por exemplo, a variável Signature da figura 2 tem um domínio com apenas dois valores (“Yes” e “No”), assim apenas um bit é necessário para representar a variável. A variável Role tem um domínio com três valores distintos, (“Researcher”, “Manager” e “User”), e assim sendo são necessários dois bits para representar a variável de negocio;
- Criar um mapeamento entre o valor de cada variável e um número binário, começado por ‘0’. Cada valor da variável de negócio tem bits *nnb* e pode ser representado por uma sequência de variáveis binárias, por exemplo, ‘ab’ ou ‘¬ab’ (o símbolo ¬ indica a negação). Por exemplo, os valores do domínio da variável Signature, “Yes” e ”No”, podem ser ‘0’ e ‘1’, respectivamente. Os valores do domínio

da variável Role, “Researcher”, “Manager” e “User” podem ser ‘00’, ‘01’ e ‘10’, respectivamente;

- Mapear os símbolos *True* ou *False* para o domínio lógico {0, 1}. O símbolo *False* é mapeado para o ‘0’ e o *True* para o 1;
- Criar uma nova tabela utilizando os dois mapeamentos descritos anteriormente.

Uma vez realizado o mapeamento, é possível extrair as funções de escalonamento da tabela de verdade. As funções extraídas são disjunções lógicas de conjunções de variáveis de negócio. Dois métodos podem ser utilizados para gerar as funções: mapas Karnaugh (Karnaugh 1953) e o método Quine-McCluskey (McCluskey 1956).

Na nossa implementação usamos o método Quine-McClusKey por ser um método particularmente útil aquando da extracção de funções de escalonamento com um grande número de variáveis de negócio. Adicionalmente, programas informáticos que implementam este algoritmo foram desenvolvidos. A utilização desta técnica aumenta o grau de automação da metodologia, sendo este um dos objectivos iniciais definidos.

A Tabela 2 mostra uma tabela de funções de escalonamento construída com base na informação presente numa TCN. A tabela é composta de três variáveis binárias: ‘a’, ‘b’ e ‘c’.

Variáveis	Tarefas	Funções de escalonamento
a,b,c	Check Form	w(a, b, c) = true
	Sign	y(a, b, c) = (¬a∧¬b) ∨ (¬a∧b∧c)
	Sign(1)	y1(a, b, c) = ¬a∧¬b
	Sign(2)	y2(a, b, c) = ¬a∧b∧c
	Notify	z(a, b, c) = (¬a∧¬b) ∨ (¬a∧b∧c) ∨ (a∧¬b∧c)
	Notify_m	z1(a, b, c) = ¬a∧¬b
	Notify_u	z2(a, b, c) = ¬a∧b∧c
	Notify_c	z3(a, b, c) = a∧¬b∧c

Tabela 1. Tabela de escalonamento construída a partir de uma tabela de casos de negócio

Quando as funções de escalonamento são disjunções de conjunções, tarefas sinónimo precisam de ser criadas. As tarefas sinónimas têm exactamente o mesmo comportamento e semântica, apenas diferem no nome. O número de disjunções na função de execução indica o número de sinónimos a serem criados para cada tarefa. Cada uma das disjunções é associada a uma tarefa sinónimo. Por exemplo, a tarefa Notify tem a função de escalonamento $(\neg a \wedge \neg b) \vee (\neg a \wedge b \wedge c) \vee (a \wedge \neg b \wedge c)$, por isso três tarefas sinónimo são criadas: Notify_m, Notify_u e Notify_c. As funções de escalonamento são decompostas nos termos “$\neg a \wedge \neg b$”, “$\neg a \wedge b \wedge c$”, e “$a \wedge \neg b \wedge c$”, e cada termo é associado a uma das tarefas sinónimo.

3.3 Identificar Estruturas de Blocos Básicas

Os sistemas de gestão de processos de negócio são centrados nos workflows, tendo uma atenção especial na gestão da lógica do fluxo. A grande maioria das linguagens de workflow são capazes de modelar blocos sequenciais, paralelos e condicionais que

são representados com estruturas standards tais como and-split, and-join, xor-split e xor-join (Aalst, Barros et al. 2000). As actividades associadas aos blocos sequenciais e paralelos são executados de uma maneira determinística, enquanto que os blocos condicionais são exemplos de uma orientação não-determinística. Os blocos condicionais indicam que o escalonamento das actividades depende da avaliação de uma condição Booleana.

A terceira fase da nossa metodologia consiste na utilização das funções de escalonamento da fase anterior para identificar os blocos sequenciais, paralelos e condicionais que irão constituir o workflow em desenvolvimento. Esta fase é composta por duas grandes etapas:

- Identificar os blocos paralelos e sequenciais associados a um workflow;
- Organizar estes blocos utilizando as estruturas condicionais.

3.3.1 Identificar Estruturas Sequenciais e Estruturas Paralelas

O objectivo da primeira etapa é identificar estruturas paralelas e estruturas sequenciais, e definir uma ordem parcial para as tarefas associadas com estas estruturas. Para completar este ponto, as seguintes passos são realizadas:

1. Criar um conjunto S de conjuntos s_i, onde cada conjunto s_i contém todas as tarefas com a mesma função de escalonamento,
2. Identificar e marcar cada conjunto s_i com a sua função de escalonamento,
3. Para cada conjunto s_i, estabelecer os blocos paralelos e sequenciais existentes, e estabelecer uma ordem parcial para as actividades.

No primeiro passo, produzimos um conjunto S de conjuntos de escalonamento s_i, onde cada conjunto s_i contém todas as tarefas com a mesma função de escalonamento. A ideia é criar um conjunto de tarefas com a seguinte propriedade: se uma actividade do conjunto s_i é escalonada em runtime, então todas as tarefas em s_i são também escalonadas. O segundo passo associa cada conjunto com um rótulo de função de escalonamento. Finalmente, o último passo estabelece os blocos sequenciais e os blocos paralelos e define uma ordem parcial para cada conjunto s_i. Cada conjunto s_i pode ser organizado utilizando blocos sequencias e/ou paralelos.

A Figura 2 mostra como a aplicação Poseidon estrutura os conjuntos s_i de tarefas. Cada conjunto está rotulado com uma função de escalonamento no canto superior esquerdo e cada conjunto contém tarefas com a mesma função de escalonamento. Para cada conjunto s_i as suas tarefas foram organizadas de forma a estabelecer os blocos paralelos e sequenciais existentes.

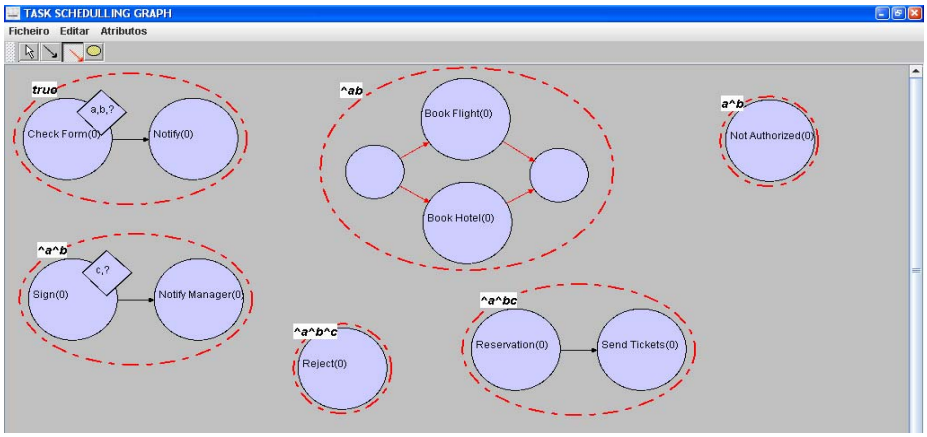


Fig. 3. Blocos paralelos e sequenciais definidos para os conjunto si criados a partir das funções de escalonamento

Estruturas condicionais não podem ocorrer para os conjuntos s_i porque o não-determinismo já foi capturado pelas funções de escalonamento.

A identificação dos blocos paralelos e sequenciais pode requerer o uso de actividades *null* (também conhecidas por actividades *dummy*). Uma actividade *null* não tem uma realização. Estas actividades podem ser utilizadas para modificar o workflow de forma a obter propriedades estruturais (e.g. bem estruturado) ou para possibilitar a modelação de procedimentos específicos de um processo de negócio.

Os dois primeiros passos podem ser automatizados, enquanto que o terceiro passo requer intervenção humana. Não obstante, acreditamos que o último passo pode ser parcialmente automatizado. Uma possível aproximação pode ser a análise das dependências dos dados e da informação entre tarefas. A dependência de dados existe entre duas tarefas se os dados de entrada de uma tarefa dependem da saída dos dados de outra tarefa. A dependência de informação existe entre duas tarefas se o conteúdo ou apresentação de uma tarefa segue o conteúdo lógico de outra tarefa. Por exemplo, vamos considerar que uma sequência de tarefas é utilizada para mostrar um contrato de negócio ao utilizador. Tendo sido identificado que diversas secções do documento necessitavam de ser aceites individualmente, foi decidido fragmentar o documento em partes. Cada parte foi associada a uma tarefa requerendo intervenção humana. Neste caso, existe uma dependência de informação entre as tarefas, desde que as tarefas porque necessitem de ser ordenadas de maneira a que o contrato seja lido em sequência lógica seguindo o documento original.

3.3.2 Identificar Estruturas Condicionais

No ponto anterior já foram identificados os blocos paralelos e os blocos sequenciais. O passo seguinte é identificar as blocos condicionais (formados por xor-splits) com base nos conjuntos s_i. O objectivo é identificar os blocos condicionais de um workflow e determinar como controlam e organizam os conjuntos anteriormente identificados (i.e. blocos paralelos e sequenciais).

Para identificar os blocos condicionais utilizamos o algoritmo de Identificação de Bloco Condicional (CBI – Conditional Block Identification) (Cardoso 2005). O algoritmo CBI pode ser visto como uma metodologia iterativs, com uma envolvente humana, para estruturar conjuntos s_i de um workflow. Este algoritmo recorre a um conjunto de assumpções e regras que são utilizados para estruturar os conjuntos de escalonamento num workflow. Depois de aplicar o algoritmo CBI, os conjuntos s_i apresentam dependências entre si representadas por transições como é mostrado na Figura 4.

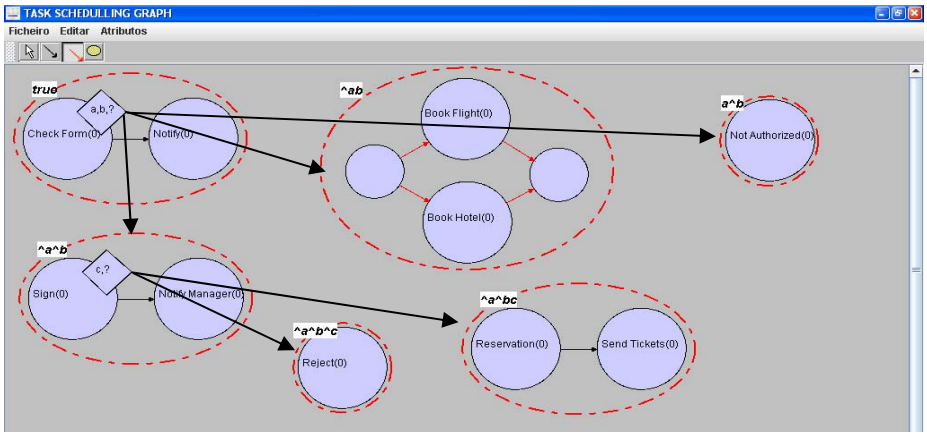


Fig. 4. Workflow com transições automaticamente criadas pelo algoritmo CBI

Contudo, vários elementos do workflow estão em falta. É aparente no exemplo que o workflow não inclui nenhuma junção dos xor-splits e o workfow tem vários pontos de saída. Ambos os problemas podem ser resolvidos pelo acerto dos xor-splits com xor-joins. Aalst (Aalst 2000) indicou a importância do equilíbrio do xor/and-splits e xor/and-joins para obter o que é chamado de "bom" workflow. Por exemplo, dois fluxos condicionais criados por um xor-split, não devem ser sincronizados por um and-join, mas por um xor-join. Equilibrar xor/and-splits pode requerer o uso de actividades *null* or *dummy*.

3.4 Limpeza, Análise e Implementação do Workflow

Na última fase, são excluídas as actividades *null* (*dummy*) e, caso necessário, o workflow poderá ser reestruturado ou modificado por motivos de clareza. Uma vez finalizadas as fases de limpeza e análise, o processo de desenho está pronto a ser implementado. O método WIDE proposto em (Casati, Fugini et al. 2002) poderá ser utilizado para esse fim. O método foca em aspectos mais técnicos e inclui a selecção do sistema de gestão de workflows alvo e o mapeamento dos workflows descrevendo um processo de negócio de alto nível na implementação do workflow. A Figura 5 ilustra a integração da metodologia Poseidon com a metodologia WISE.

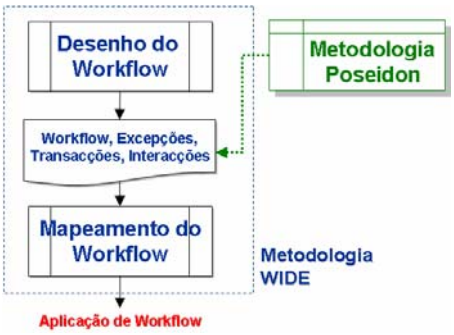


Fig. 5. Integração da metodologia Poseidon com a metodologia WISE

Na fase de desenho do workflow, a metodologia WIDE fornece conceitos que permitem implementar o workflow resultante em diferentes sistemas de gestão de workflow, tendo em conta as diferentes características dos sistemas. No fim da fase de desenho o seguinte resultado é obtido: um esquema do workflow, a especificação das excepções e transacções, e especificação das interacções com aplicações externas. A fase de desenho do workflow da metodologia WIDE pode ser substituída ou complementada pela metodologia Poseidon. A metodologia Poseidon é mais poderosa pois permite a o desenho semi-automático dos esquemas de workflows.

4 Implementação

Foram consideradas várias linguagens de programação durante a fase de análise da aplicação Poseidon. O MS Visual Basic, linguagem C e Java. Mas desde cedo tornou-se evidente que a implementação multiplataforma e a portabilidade do código eram aspectos que se revestiam de grande importância tendo influenciado a nossa escolha pela selecção da linguagem Java. Para o desenho gráfico dos workflows consideramos inicialmente usar os componentes Swing que a linguagem Java oferece ou usar uma ferramenta direccionada para a implementação rápida e prática de aplicações que manipulem grafos. A implementação da aplicação com o uso dos componentes Swing iria revelar-se trabalhosa e demorada, uma vez que todas funcionalidades associadas com o desenho de workflows teriam de ser programadas. Tendo em conta que os requisitos da aplicação Poseidon eram relativamente standards do ponto de vista da interface gráfica, optamos por analisar as ferramentas JGraph (www.jgraph.com) e JHotDraw (http://www.jhotdraw.org/). Finalmente, seleccionamos o JHotDraw pois a sua arquitectura foi desenvolvida com base num conjunto conhecido de *patterns* que forneciam soluções arquitectónicas para as funcionalidade que necessitávamos para a aplicação Poseidon.

4 Conclusões

Apesar da investigação realizada no sentido de evoluir os sistemas de gestão de workflow, o trabalho realizado sobre ferramentas, metodologias e métodos para suportar a fase de desenho dos workflows é praticamente inexistente.

O desenvolvimento de métodos e ferramentas adequadas é de grande importância para garantir que os workflows sejam construídos de acordo com as especificações iniciais. Infelizmente, é reconhecido que apesar da difusão dos sistemas de gestão de workflow, as ferramentas para suportar o desenho de workflows continuam a ser uma lacuna. Neste artigo, é descrito uma aplicação, chamada Poseidon, para assistir os analistas de processos durante as entrevistas com o pessoal administrativo, gestores e empregados em geral, de forma a possibilitar o desenho semi-automático de workflows.

A aplicação Poseidon apresentada foi utilizada com sucesso para desenhar workflows administrativos de média dimensão numa empresa do sector aeronáutico. Existe a convicção de que a aplicação é igualmente apropriada para desenhar workflows de grande escala, bem como o facto da aplicação representar um passo em frente na modelação de processos de negócio.

Referências

Aalst, W. M. P. v. d. (1998). "The Application of Petri Nets to Workflow Management." The Journal of Circuits, Systems and Computers **8**(1): 21-66.

Aalst, W. M. P. v. d. (2000). Workflow Verification: Finding Control-Flow Errors Using Petri-Net-Based Techniques. Business Process Management: Models, Techniques, and Empirical Studies. W. M. P. v. d. Aalst, J. Desel and A. Oberweis. Berlin, Springer-Verlag. **1806:** 161-183.

Aalst, W. M. P. v. d., A. P. Barros, et al. (2000). Advanced Workflow Patterns. Seventh IFCIS International Conference on Cooperative Information Systems, September 2000.Vol**:** pp. 18-29,

Aalst, W. M. P. v. d. and A. H. M. t. Hofstede (2003). YAWL: Yet Another Workflow Language (Revised Version). Brisbane, Queensland University of Technology2003.

ARIS (2005). ARIS Design Platform. http://www.ids-scheer.com/.

BPML (2004). Business Process Modeling Language. **2004**.

BPMN (2005). Business Process Modeling Notation - http://www.bpmn.org/.

Cardoso, J. (2005). "Poseidon: A framework to assist Web process design based on business cases." International Journal of Cooperative Information Systems (IJCIS)(accepted for publication).

Cardoso, J., J. Miller, et al. (2004). "Modeling Quality of Service for workflows and web service processes." Web Semantics: Science, Services and Agents on the World Wide Web Journal **1**(3): 281-308.

Cardoso, J. and A. Sheth (2003). "Semantic e-Workflow Composition." Journal of Intelligent Information Systems (JIIS). **21**(3): 191-225.

Casati, F., M. Fugini, et al. (2002). "WIRES: a Methodology for Designing Workflow Applications." Requirements Engineering Journal **7**(2): 73-106.

COSA (2005). COSA Workflow - http://www.cosa-bpm.com/.

Karnaugh, M. (1953). "The Map Method for Synthesis of Combinational Logic Circuits." Transaction IEEE **72**(9): 593-599.
Kochut, K. J. (1999). METEOR Model version 3. Athens, GA, Large Scale Distributed Information Systems Lab, Department of Computer Science, University of Georgia.
McCluskey, E. J. (1956). "Minimization of Boolean functions." Bell System Technical Journal **35**(5): 1417-1444.
McCready, S. (1992). There is more than one kind of workflow software. Computerworld. **November 2:** 86-90.
Miller, J., A. Sheth, et al. (1996). "CORBA-based Run Time Architectures for Workflow Management Systems." Journal of Database Management, Special Issue on Multidatases **7**(1): 16-27.
Sheth, A., D. Georgakopoulos, et al. (1996). Report from the NSF Workshop on Workflow and Process Automation in Information Systems. Athens, GA, Deptartment of Computer Science, University of Georgia.
Sheth, A. P., W. v. d. Aalst, et al. (1999). "Processes Driving the Networked Economy." IEEE Concurrency **7**(3): 18-31.
TIBCO (2002). TIBCO InConcert, TIBCO.
WS-BEPL (2005). Business Process Execution Language for Web Services, http://www-128.ibm.com/developerworks/library/specification/ws-bpel/.