Flexibilidade em Workflows Baseados em Restrições

Fábio de Lima Bezerra[1] **, Jacques Wainer**[1]

[1]Instituto de Computação – UNICAMP
Caixa Postal 6176 – 13083–970 Campinas, SP

{fabio.bezerra,wainer}@ic.unicamp.br

Abstract. *In some domains as the software engineering, software quality and helthcare the production workflow systems are not adjusted because the processes defined in this type of workflow are very rigid, demanding of the designer a complete definition of the processes, which is complicated or impossible in these domains. We present in this work a new approach of process definition, the use of workflows based on overriding constraints. Moreover, this work will present the Tucupi server, a prototype of a workflow system that supports such new approach.*

Resumo. *Em alguns domínios como a engenharia de software, qualidade e medicina os sistemas de workflow de produção não são adequados, pois os processos definidos neste tipo de workflow são muito rígidos, exigindo do projetista uma definição completa dos processos que serão executados nestes sistemas. Apresentar uma definição completa do processo nesses domínios é muito complicado ou impossível. Apresentamos neste trabalho um nova abordagem de definição de processos, o uso de* restrições violáveis. *Através do uso das restrições poderemos definir o que chamaremos de workflows parciais, ou seja, workflows que não possuem uma definição completa e que podem ser dinâmicamente planejados. Além disso, este trabalho apresentará o servidor Tucupi que é um protótipo de um sistema de workflow que suporta workflows a definição de workflows parciais.*

1. Introdução

O termo workflow muitas vezes é confundido com um sistema computacional, software ou aplicação. Apesar de ser utilizado com este sentido, o workflow representa um processo que suporta automatização computacional, enquanto que os sistemas de gerência de workflow (ou WFMS) são os sistemas que permitem a definição, execução e monitoração de processos (workflows). Na execução de uma instância de workflow uma série de atividades é executada por pessoas ou automaticamente, de acordo com a definição do processo. Em alguns domínios, a definição do processo é criada antes de seu uso, embora a maioria das definições utilize formalismos que permitem alguma flexibilidade, como por exemplo a execução condicional de caminhos (pré-definidos) diferentes baseados na escolha do usuário ou em alguns dos dados da instância do workflow. Neste caso, dizemos que o workflow é flexível porque uma única definição do workflow permite caminhos distintos de execução, apesar de previamente definidos.

Em domínios como a engenharia de software e medicina o projetista não conhece a estrutura de execução do processo antes de sua execução, ou seja, as atividades e a ordem de execução das atividades do processo é desconhecida. Portanto existe a necessidade de se implementar nesses domínios um forma alternativa de flexibilidade. Para isso usaremos o que chamamos de *workflows parciais*, que é um workflow condicional e parcialmente definido. Assim, consideraremos a flexibilidade como a capacidade de um workflow ser dinamicamente definido.

Um workflow parcial é formado pelas atividades que podem ser executadas nesse workflow e pelas restrições entre as atividades desse workflow. Os workflows parciais podem incluir em sua definição atividades que devem ser executadas, mas admitem uma execução desordenada. Por exemplo, a construção de vários componentes independentes de um grande projeto de software. Outra característica dos workflows parciais é a existência de atividades cuja execução é condicionada a execução de outra(s) atividade(s) antes. Por exemplo, caso uma interveção cirúrgica seja requisitada o paciente deve assinar um termo de consentimento de transfusão de sangue no caso de haver a necessidade.

As restrições são apresentadas como instrumento de definição de um workflow. Acreditamos que o workflow definido como um conjunto de restrições possibilita uma execução menos rígida, pois uma restrição não define uma estrutura de execução, mas uma declaração de dependência entre as atividades envolvidas na declaração da restrição. Ou seja, uma restrição pode ser verdadeira em diferentes cenários de execução.

Um workflow também atende os requisitos de flexibilidade quando dispõe de um mecanismo de tratamento de exceções, pois acrescenta, na definição do workflow, caminhos alternativos de execução que são percorridos em situações excepcionais. Para atender este requisito de flexibilidade usaremos o conceito de *restrições violáveis*, ou seja, restrições que podem ser violadas em condições excepcionais.

Este artigo está estruturado como segue: a próxima seção discute as diferentes abordagens que os sistemas de workflow podem assumir, destacando a abordagem utilizada neste trabalho; a Seção 3 apresenta as restrições como instrumento de definição de um workflow que suporta flexibilidade; a Seção 4 faz uma breve apresentação do Tucupi, um protótipo de um sistema de workflow que implementa as idéias apresentadas neste artigo; a Seção 5 apresenta alguns trabalhos correlatos; e por fim apresentamos nossas conclusões na Seção 6.

2. Planejamento e execução

Os sistemas de workflow usados em ambientes de negócios geralmente utilizam apenas workflows cuja definicão é completa ou total, ou seja, o workflow possui todas as atividades que serão executadas bem como a ordem de execução destas atividades. Nestes workflows, quando uma atividade é concluída o sistema consegue determinar quais atividades precisam iniciar em seguida. Ou seja, a definição ou planejamento do workflow reflete a execução do mesmo. Neste caso dizemos que o workflow cumpre o papel de *despachante* das atividades.

No caso dos workflows parciais, nem sempre é possível pro sistema determinar qual a próxima atividade executar. Assim, em domínios de processos fracamente definidos, normalmente existe uma pessoa ou pessoas que chamaremos de *controladores* do processo. Tais pessoas decidem que atividade deve ser iniciada no caso. Tal atividade será chamada de *atividade objetivo*. Por exemplo, o gerente de projeto do software financeiro (controlador do caso) decide que a implementação do componente fiscal (atividade objetivo) deve ser iniciada.

A execução de uma atividade solicitada pelo controlador do workflow pode ser condicionada a execução de outras atividades antes. Neste cenário o workflow funciona como um *ajundante*, pois o workflow ajudará o controlador sobre as implicações da execução da atividade objetivo. Tais implicações normalmente involvem a necesidade de executar atividades tanto antes quanto depois da atividade objetivo. As atividades que devem ser executadas ou antes ou depois da atividade objetivo são chamadas de *atividades obrigratórias*. Por exemplo, a implementação de um componente deve ser precedida pela documentação dos requisitos e criação dos diagramas de classes e objetos. Assim, no cenário do workflow como ajudante, a execução da atividade objetivo pode ser proibida de ser executada ou atrasada, aguardando a execução das atividades que são pré-condições dela.

O planejamento dos workflows como um despachante é fixo e completo, enquanto que o planejamento dos workflows como um ajudante é dinâmico, pois a definição do workflow é completada durante a execução. Ou seja, no cenário do workflow como ajudante, o controlador da instância de workflow possui bastante autonomia na execução. É o controlador que determina as atividades que precisa executar, definindo dinâmicamente o fluxo de execução que julgar conveniente (no caso das atividades que não possuem restrições).

O controlador pode decidir que algumas das atividades obrigatórias de uma restrição não pode ou não deveria ser executada. Neste caso, o controlador verifica no sistema de workflow se ele possui autoridade suficiente para violar a restrição.

3. Restrições: uma solução para flexibilidade

Sabemos que para alguns domínios como a engenharia de software, qualidade de software e medicina os workflows de produção não são adequados porque utilizam uma definição muito rígida do processo. Assim,

propomos o uso das restrições na definição dos workflows a fim de atender os requisitos de flexibilidade desses domínios.

3.1. Especificação

Apresentaremos a seguir uma especificação do modelo de definição de restrições. Para tanto utilizaremos uma linguagem simplificada que fornece os atributos necessários para representar uma restrição, que são:

Rule .
Este atributo indica o nome da restrição, que deve ser único na definição do workflow.

Target .
Este atributo indica para que atividade a restrição se aplica, ou seja, a atividade cujo o usuário do sistema deseja executar sobre restrições.

Precondition .
Este atributo indica as atividades que devem ser executadas antes de executar a atividade indicada pelo atributo *Target*, ou seja, representa as restrições de execução da atividade objetivo ou atividades obrigatórias.

Postcondition .
Este atributo indica as atividades que devem ser executadas após a execução da atividade indicada pelo atributo *Target*, ou seja, representa as restrições de execução da atividade objetivo ou atividades obrigatórias.

Para entermos melhor a definição de cada um dos atributos listados acima considere a seguinte restrição de um processo fictício de desenvolvimento de software:

”A implementação de um componente de software não pode ser realizada sem que exista uma especificação formal e um diagrama de classes desse componente. Após a construção o componente deve ser testado.”

Esta restrição de construção do componente poderia ser representada como uma restrição da seguinte maneira:

```
rule: C01
   target: "implementar componente"
   precondition: "desenvolver especificação formal"
   precondition: "criar diagrama de classes"
   postcondition: "testar componente"
```

Através das restrições poderemos implementar o que chamamos de workflow parcial, ou seja, um workflow cuja definição é limitada a especificação de pequenos trechos de execução. Os trechos de execução são formados a partir da definição das restrições que compõem a definição do workflow. Onde temos:

- a restrição é definida para uma atividade chamada atividade objetivo;
- cada restrição é formada por um conjunto não vazio de atividades obrigatórias, ou seja, atividades que devem ser executadas antes (pré-condições) e/ou depois (pós-condições) da execução da atividade objetivo;
- Um trecho de execução é formado pela cadeia gerada a partir das atividades que são pré e pós condições de uma atividade objetivo.

Outras construções podem ser realizadas com a linguagem acima, por exemplo:

Disjunção de pré-condições e pós-condições .
As pré e pós-condições formam uma conjunção de restrições, mas para representarmos uma disjunção de pré e/ou pós-condições, basta incluir na definição de uma pré-condição as atividades que formam a disjunção de pré-condições separadas pelo conectivo OR:

```
rule: c01
   target: E
   precondition: A or B
   precondition: C or D
   postcondition: F or G
```

para E ser executada A ou B deve ser executada antes e C ou D também deve ser executada antes. Após E ser executada F ou G deve ser executada.

Restrição de não execução de uma atividade .

As pré e pós-condições indicam que a atividade deve ser executada. Para indicar uma restrição de que a atividade não deve ser executada, basta utilizar o operador NOT:

```
rule: c01
   target: B
   precondition: not A
```

para B ser executada A não deve ter sido executada antes.

Atributo *parcondition* .

Este atributo indica as atividades que devem ser executadas antes ou depois da execução da atividade indicada pelo atributo *Target*. Também representa as atividades obrigatórias.

```
rule: c01
   target: B
   parcondition: A
```

para B ser executada A deve ter sido executada antes ou deve ser executada depois.

3.2. Restrições violáveis

As exceções nos sistemas de workflow podem ser de dois tipos, as provocadas por mal funcionamento de algum componente de software ou hardware do sistema; e as provocadas por desvios do fluxo de execução planejado pelo projetista do workflow (do processo). As exceções que estamos interessados são as que forçam um novo fluxo de execução no workflow, pois os sistemas de workflow necessitam de um mecanismo de tratamento desses desvios.

O tratamento de exceções em um workflow é considerado um suporte a flexibilidade, pois permite que um fluxo alternativo de execução seja tomado. Por exemplo, quando um paciente encontra-se inconsciente, o médico pode violar a restrição que obriga o paciente de assinar um termo de consentimento de transfusão de sangue antes de uma intervenção cirúrgica.

As restrições violáveis são uma extensão a proposta de uso das restrições na definição de um workflow, como apresentamos acima. O uso das restrições violáveis ofere aos sistemas de workflow uma mecanismo de tratamento de exceções. Entretanto, a simples possibilidade de violar uma restrição não nos parece uma solução, por isso sugerimos o acoplamento do sistema de workflow com o modelo WRBAC [Wainer et al., 2001, withheld, 2003].

WRBAC é uma extensão do mecanismo de controle de acesso RBAC (Role-Based Access Control, [Sandhu et al., 1996, Ferraiolo et al., 1995]) para sistemas de workflow. WRBAC adiciona ao RBAC o conceito de *caso* (instância de workflow), e define cada atividade em um workflow como um privilégio. Com o conceito de caso é possível modelar restrições dinâmicas, que são as restrições que se aplicam a uma instância de workflow.

Utilizamos o WRBAC para modelarmos a violação de uma restrição como um privilégio. Desta forma estamos implementando um mecanismo de violação controlado das restrições, tal que somente será possível violar uma restrição da definição do workflow o usuário que possuir o privilégio de violação desta restrição.

4. O Servidor Tucupi

O servidor Tucupi é um protótipo de um sistema de workflow que suporta o uso de restrições na definição de um workflow. O servidor Tucupi foi implementado em Prolog e funciona como um serviço que utiliza sockets como mecanismo de comunicação. Assim, o servidor escuta uma porta TCP/IP onde clientes podem se conectar e executar os serviços que o servidor disponibiliza através de sua interface pública.

O servidor Tucupi armazena as seguintes informações:

- a hierarquia organizacional, os papéis e os usuários (WRBAC);
- as restrições de execução (WRBAC);
- as restrições de ordem de cada workflow;

- as regras de ativação de cada workflow;
- o estado das instâncias de workflow.

As restrições de ordem e as regras de ativação de cada workflow representam a definição de um workflow, que são usadas durante a execução de uma instância de workflow. Entretanto, as regras de ativação não são incluídas pelo projetista do processo, mas derivadas a partir de uma ferramenta de definição.

4.1. Derivação de uma restrição

A derivação de uma restrição é um mecanismo de transformação de uma restrição. Uma restrição indica o que *não* é permitido fazer, portanto podemos dizer que as restrições são regras negativas [Wainer, 2000]. Entretanto, os sistemas de workflow precisam, além das restrições, de regras que indicam o que fazer. Chamaremos as regras que dizem o que fazer de regras positivas. A partir da declaração de uma restrição é possível inferir, além da restrição em si, regras positivas. Portanto, a derivação de uma restrição gera as seguintes classes de regras:

Regras de ativação. As regras de ativação indicam o que fazer e quando fazer.
Restrições de ordem. As restrições de ordem indicam qual a ordem de execução entre duas atividades, ou seja, a relação de dependência.

Por exemplo, considere a restrição abaixo:

```
rule: c01
    target: B
    precondition: A
    postcondition: C
```

A partir da restrição acima podemos derivar as seguintes regras em prolog [Bratko, 1990]:

```
constraint(c01) :- not(override(c01)), %Restrição
                   not(start(b) -> end(a);true).

next(b) :- not(start(b)), end(a). %Ativação
next(c) :- not(start(c)), end(b). %Ativação
```

5. Trabalhos correlatos

Existem muitos trabalhos relacionados ao provimento de flexibilidade nos sistemas de workflow, mas muitos destes trabalhos seguem linhas de pesquisas diferentes das adotadas aqui. Entre as abordagens utilizadas podemos citar as que tratam o problema da mudança dinâmica na definição de um processo e as conseqüências dessas mudanças nas instâncias de workflow ativas [Ellis and Keddara, 2000, Aalst, 2001]. Uma segunda abordagem refere-se ao entendimento e desenvolvimento de um mecanismo de tratamento de exceções, ou seja, refere-se ao comportamento do sistema de workflow durante um desvio excepcional do fluxo de execução de uma instância de workflow.

Poucos trabalhos foram encontrados na literatura sobre o uso de restrições na definição de workflows. Entre os trabalhos encontrados destacamos [Mangan and Sadiq, 2002b, Mangan and Sadiq, 2002a], que é o que mais se aproxima de nossa pesquisa. Neste trabalho, um workflow é definido através de um conjunto de segmentos de workflows e de restrições de seleção e construção. A definição completa do workflow ocorre apenas em tempo de execução, através da junção dos segmentos de workflow, que satisfazem as restrições definidas para o workflow.

Paul Dourish et al. em [Dourish et al., 1996] descreve o *Freeflow*, o protótipo de um sistema de workflow que utiliza restrições em seu modelo de definição de workflow. Neste trabalho Dourish apresenta as restrições como uma forma declarativa de definir dependências entre as atividades de um workflow e as ações do usuário são comparadas com a definição. No *Freeflow* o usuário tem total controle sobre as ações que deseja tomar, mesmo no momento de violar uma restrição. O sistema apenas alerta o usuário sobre a existência da restrição e questiona-o sobre prosseguimento da ação na presença da restrição. Outra limitação desse trabalho refere-se a inexistência de um mecanismo de ativação de tarefas.

O trabalho em [Wainer, 2000] também tem similaridades. Este trabalho utiliza uma lógica temporal para representar uma ordenação entre as atividades de um workflow. Este trabalho também apresenta um mecanismo de ativação de atividades através do uso de uma lógica não-monotônica que infere quais atividades devem ser iniciadas imediatamente.

6. Conclusão

Apresentamos neste trabalho o uso das restrições como uma proposta diferente de definir/modelar um workflow. Percebemos que o uso das restrições proporciona uma definição de workflow mais flexível, permitindo a uma única definição de workflow diferentes estruturas de execução. Além disso, as restrições violáveis estendem o conceito de flexibilidade, pois são uma solução para o tratamento de exceções nos sistemas de workflow.

Workflows com limites temporais entre atividades não são comuns na literatura ([Eder and Panagos, 2000, Marjanovic and Orlowska, 1999] são alguns exemplos). Talvez nas aplicações de negócios tais limites temporais não sejam comum e por isso não tenham sido contemplados na literatura de workflow. Entretanto, em áreas como a medicina torna-se claro o uso de restrições temporais. Este trabalho ainda encontra-se em estágio inicial a respeito da definição e manipulação de restrições temporais.

Referências

Aalst, W. M. P. V. D. (2001). Exterminating the dynamic change bug: A concrete approach to support workflow change. *Information Systems Frontiers*, 3(3):297–317.

Bratko, I. (1990). *Prolog Programing for Artificial Inteligence*. International Computer Science Series. Addison-Wesley. 2nd Edition.

Dourish, P., Holmes, J., MacLean, A., Marqvardsen, P., and Zbyslaw, A. (1996). Freeflow: mediating between representation and action in workflow systems. In *Conference on Computer Supported Cooperative Work CSCW96*. ACM.

Eder, J. and Panagos, E. (2000). Managing time in workflow systems. In Fischer, L., editor, *Workflow Handbook 2001*, pages 109–132. Future Strategies INC.

Ellis, C. and Keddara, K. (2000). Ml-dews: Modeling language to support dynamic evolution within workflow systems. *Computer Supported Cooperative Work*, 9(3-4):293–333.

Ferraiolo, D., Cugini, J., and Kuhn, R. (1995). Role based access control: Features and motivations. In *Annual Computer Security Applications Conference*, IEEE Computer Society Press, New Orleans, Louisianna.

Mangan, P. and Sadiq, S. (2002a). On building workflow models for flexible processes. In *The Thirteenth Australasian Database Conference ADC2002*, Melbourne, Australia.

Mangan, P. J. and Sadiq, S. (2002b). A constraint specification approach to building flexible workflows. *Journal of Research and Practice in Information Technology*.

Marjanovic, O. and Orlowska, M. (1999). On modeling and verification of temporal constraints in production workflows. *Knowledge and Information Systems*, 1(2).

Sandhu, R., Coyne, E., Feinstein, H., and Youman, C. (1996). Role-based access control models. *IEEE Computer*, 29(2):38–47.

Wainer, J. (2000). Logic representation of processes in work activity coordination. In *Proceedings of the ACM Symposium on Applied Computing, Coordination Track*, volume 1, pages 203–209. ACM, ACM Press.

Wainer, J., Barthelmess, P., and Kumar, A. (2001). *WRBAC* - A workflow security model incorporating controlled overriding of constraints. Technical report, Universidade Estadual de Campinas.

withheld, N. (2003). W-RBAC: a workflow secutiry model incorporating controlled overriding of constraints. submitted.