Suporte a Adaptação Dinâmica em Sistemas de Gerência de Workflow usando Técnicas de Inteligência Artificial

Lucas Bucci da Silveira[1], Carlos Roberto Lopes[1]

[1] Faculdade de Computação, Universidade Federal de Uberlândia, 38.400-902, Uberlândia, Brasil
{lucas, crlopes}@facom.ufu.br

Abstract. Workflow systems are related with coordination and collaboration of activities supported by human or non-human resources. Current workflow management systems lack support for dynamic and automatic workflow adaptations. However, this functionality is a major requirement for next–generation workflow systems to provide sufficient flexibility to cope with unexpected failure events. In this paper we show how a workflow management system equipped with an execution mechanism based on a petri net model guided by situated control rules can deal with exceptions. Such control rules are synthesized by the use of a planning algorithm. With the combination of conditional planning and replanning it is possible to anticipate exceptions or treat them during execution.

Keywords: Artificial Intelligence, Workflow Management Systems, Artificial Intelligence Planning, Exceptions.

1 Introdução

Sistemas gerenciadores de *Workflows* (*SGWfs*) representam uma infra-estrutura tecnológica que gerencia eficientemente processos de negócios. O termo *workflow* é aplicado freqüentemente para designar processos de negócios; sistemas que executam e automatizam processos, ou aplicações que simplesmente suportam a colaboração e coordenação de pessoas e sistemas para execução de um processo. Organizações utilizam atualmente os *SGWfs* para gerenciar inúmeras aplicações, tais como, requisição de seguros, empréstimos bancários, procedimentos administrativos, e até mesmo em processos de auxílio no tratamento de doenças [1]. Alguns dos problemas em que as tecnologias de *workflow* podem ser utilizadas possuem características dinâmicas. Isto significa que não é possível determinar antecipadamente todas as atividades necessárias para a conclusão do processo. As ocorrências de situações indesejadas em um sistema de *workflow* são chamadas de exceções. Portanto *SGWfs* que atuem em ambientes dinâmicos necessitam oferecer um tratamento adequado para as diversas exceções que podem ocorrer durante a execução de um *workflow*.

XXXIV Conferencia Latinoamericana de Informática (CLEI 2008), pp. 220-229, 2008.

A utilização de *workflows* não se restringe apenas a empresas. W*orkflows* para pesquisa científica têm sido empregados para especificar e coordenar a execução de experimentos que envolvem participantes em locais distintos. Eles permitem a representação e execução de atividades que usam dados e ferramentas heterogêneas.

O interesse em aplicar técnicas de inteligência artificial (*IA*) em sistemas tradicionais para solucionar problemas tem crescido nos últimos anos. Existe um interesse especial nas técnicas que envolvem planejamento (*AI Planning*), as quais já são bastante utilizadas em áreas especializadas como: missões espaciais, robótica, planejamento de missões militares, controle de elevadores etc [2], [3]. Além destas áreas existe o entendimento que muitas outras podem tirar proveito da possibilidade de automação oferecida pela aplicação de uma tecnologia de *AI Planning*. Nesta linha encontra-se o *PLANET* [3], uma rede de excelência, cujo objetivo é fomentar o desenvolvimento e integração desta tecnologia com diversas áreas. Esta entidade foi organizada em unidades menores, chamadas *Technical Coordination Unit (TCU)*, responsáveis por conduzir o desenvolvimento em cada Área. Uma das linhas de trabalho promove a aplicação efetiva das técnicas de planejamento e escalonamento aos sistemas de *workflow*.

Neste trabalho implementamos um mecanismo de execução adaptativa de *workflows* baseado nas propriedades da rede de planos [4]. A partir do modelo de *workflow* gerado pelo planejador *Metaplan* [5], regras de controle situadas (*RCS*) são extraídas para cada passo do modelo e a partir dessas regras o mecanismo de execução pode decidir apropriadamente entre quais atividades executar para que o processo possa ser concluído com sucesso. Para ambientes dinâmicos, novas situações podem ocorrer as quais não foram criadas regras para guiar a execução. Nestes casos, executar uma atividade aleatória não irá garantir o sucesso do processo. Para tornar o mecanismo de execução adaptável são utilizadas técnicas de planejamento para construir novos caminhos a partir de pontos no qual não existam *RCS*. Os novos caminhos encontrados são transformados em regras que realimentam o mecanismo de execução. Através desse mecanismo de execução podemos fazer a execução adaptativa dos modelos de *workflows*.

A estruturação das demais seções deste artigo é apresentada a seguir. As seções 2 e 3 apresentam respectivamente definições sobre as áreas de *Workflow* e Planejamento para uma melhor compreensão do trabalho realizado. A seção 4 apresenta o desenvolvimento do trabalho e na seção 5 alguns trabalhos relacionados são comparados com essa abordagem. A conclusão é apresentada na seção 6.

2 Workflow

A definição para *workflow* dada pela *Workflow Management Coalition Specification* (*WfMC*) é a seguinte: "*Workflow é a automação de processos de negócios, no todo ou em parte, onde documentos, informações ou atividades são passadas de um participante para outro, de acordo com um conjunto de regras*" [6]. Sistemas Gerenciadores de *Workflows* (*SGWfs*) são ferramentas capazes de modelar, analisar e executar modelos de *workflow*.

A modelagem de um *workflow* inicia-se na tradução do seu modelo real para o ambiente computacional, gerando uma formalização através de técnicas de modelagem apropriadas. O resultado é um modelo ou representação do processo a ser executado pelo mecanismo de execução do *SGWfs*. Assim a modelagem tem como objetivo produzir uma abstração de um processo que serve como base para especificar um *workflow*.

O elemento principal em um *workflow* são as atividades. As atividades são as unidades de trabalho do *workflow*. As atividades podem ser encadeadas sob três diferentes formas: *seqüencialmente*, em *paralelo* e *condicional*. Seqüencialmente significa que tão logo uma atividade seja executada a tarefa seguinte é executada. Atividades executando em paralelo são ativadas simultaneamente e encaminhadas para seus respectivos executores. Não necessariamente irão ser concluídas no mesmo tempo, já que podem seguir critérios diferentes e ou demandar operações distintas. Em um dado momento estes fluxos paralelos irão convergir em um fluxo seqüencial ou na finalização do processo. O encadeamento condicional surge quando a próxima tarefa a ser executada baseia-se em uma decisão.

A execução de um processo dá-se pela instanciação de sua definição, ou seja, a associação do modelo de um processo para casos particulares. Nesta etapa as variáveis são instanciadas (ou recebem valores) e recursos são associados às atividades para execução.

3 Planejamento

O planejamento em *IA* envolve determinar um conjunto ordenado de ações (designado por *plano*) que quando executadas por um ou mais agentes a partir de um estado inicial que satisfaça a circunstâncias dadas, resulte num estado final que satisfaça uma dada meta ou conjunto de metas. As definições de planejamento apresentadas no relatório do *PLANET* complementam sutilmente esta definição, introduzindo o conceito de processo. Um processo corresponde a um conjunto ordenado de atividades não instanciadas. Um plano é interpretado como uma seqüência de atividades para atingir um dado objetivo, isto é, um processo instanciado [3]. Estas definições são complementares, e introduz o processo como o conjunto de todos os planos válidos.

Formalmente, o planejamento é definido através da tripla (*A, I, G*) cujos elementos são o conjunto de ações (A), o estado inicial (I), e o estado final ou meta (G). Seja *P* o conjunto de todas as proposições que representem fatos no mundo. O estado corrente, ou mundo, é designado por *w* e representa o subconjunto de proposições satisfeitas em *P* tal que $w \subseteq P$ no mundo. Para representar um problema de planejamento clássico geralmente se utiliza uma linguagem de representação baseada no algoritmo de planejamento (ou planejador) *STRIPS* [7].

Em *STRIPS*, as ações são triplas do tipo (*pre(a), add(a), del(a)*) cujos elementos pertencem ao conjunto de proposições e correspondem respectivamente às precondições e aos efeitos das ações – esta última através das listas de adição e remoção. Uma ação *a* é aplicável em *w* if $w \supseteq pre(a)$. Aplicar *a* em *w*, substitui, *w* por *w'* tal que $w' = w - del(a) + add(a)$. Assume-se que $del(a) \cap add(a) = \{\ \}$.

3.1 Metaplan

O trabalho que será apresentado utilizou-se do planejador *Metaplan*. Na modelagem de processos de *workflows* os conceitos de pré-condição e efeito de planejamento não estão explicitamente presentes, por isso o *Metaplan* utiliza dois atributos de extensão para representar essa informação em cada atividade, que são respectivamente *Pré-Condição* e *Efeito*. O planejador *Metaplan* [5], se beneficia da evolução dos planejadores para realizar planejamento condicional. Utilizando atualmente o planejador *FF* [8] como gerador de planos seqüenciais, os algoritmos do *Metaplan* resolvem problemas de planejamento condicional, no qual um plano não é apenas uma seqüência de ações e sim várias ramificações possíveis de ações para atingir a meta. Além da geração de planos condicionais o planejador *Metaplan* é capaz de gerar ações que podem ser executadas em paralelo. Desta forma é possível gerar automaticamente as principais construções que aparecem num *workflow* apresentados na seção 2.

4 Gerenciando a Execução de Workflows na presença de Exceções

Em ambientes onde ocorram constantes mudanças que levem a situações não previstas (designadas por exceções), um mecanismo de adaptação do *workflow* é necessário para que a sua execução tenha sucesso. O mecanismo de execução adaptativo é um componente do *SGWf*.

Para fins de tratamento de exceções este trabalho propõe o uso de regras de controle situadas, as quais também permitem guiar a execução do *workflow*. Adicionalmente, este trabalho também propõe o uso de planejamento durante a execução, o que possibilita a criação de novos caminhos para as situações não previstas. O algoritmo desenvolvido será apresentado a seguir juntamente com as definições necessárias para melhor entendimento do trabalho.

4.1 Rede de Planos

Uma rede de planos é um tipo de rede de *Petri* conhecida como Condição/Evento [4]. Trata-se de um grafo direcionado construído a partir de dois nodos chamados de *condições* e *operadores*. As *condições* são fatos que determinam a situação do mundo. Os *operadores* são atividades que podem ser aplicadas em uma dada situação. Os arcos entre os nodos descrevem relações de permissões e efeitos, ou seja, estabelece as pré-condições para os operadores serem aplicados e os efeitos da aplicação dos operadores.

O estado sucessor à aplicação de um *operador* é obtido removendo todas as pré-condições do *operador* e adicionando seus efeitos. A tabela 1 apresenta a modelagem de um problema onde um fabricante recebe o pedido para produzir mercadorias a serem entregues, e para isso precisa verificar de que forma adquirir a matéria prima para produzir a mercadoria. Os atributos *Pré-Condição* e *Efeito* são os atributos de extensão propostos por [5] para adicionar nas atividades os conceitos de planejamento. Estes atributos definem as transições entre as atividades.

Tabela 1. Modelagem do processo de workflow de fabricação de mercadorias.

Atividade	Pré-Condição	Efeito
Pedido	n_order	ret_i v ret_s
Retirar do Estoque	ret_i	a_order
Retirar do fornecedor Favorito	ret_s	a_order
Montar Mercadoria	a_order	assemb $\wedge$ cont
Contabilizar Mercadoria	cont	d_cont
Expedir Mercadoria	assemb	exp_merc

A partir da rede de planos é possível fazer projeções. Projeções são construídas aplicando-se repetidamente os *operadores* a partir de uma *condição*. O modelo de *workflow* gerado previamente por meio do *metaplan* constitui-se, na realidade, de uma seqüência de situações obtidas a partir de uma dada situação inicial. Dessa forma o resultado do processo de planejamento é também uma projeção da rede de planos. A figura 1 apresenta o modelo de *workflow* para a tabela 1. A projeção é importante no sentido de estabelecer regras que permitam especificar para uma dada situação quais operações podem ser executadas. Tais regras serão descritas na próxima subseção.

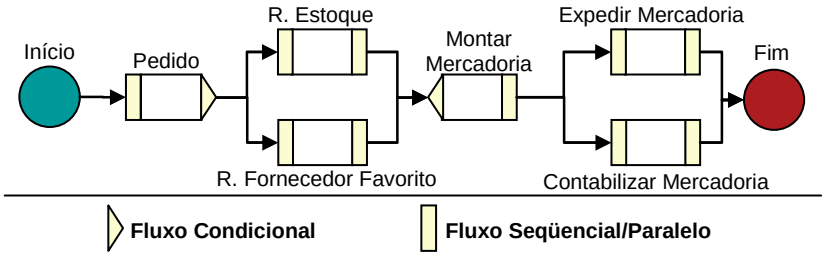

Fig. 1. Modelo de *workflow* da tabela 1.

4.2 Regras de Controle Situadas

Numa rede de planos, para uma dada situação do processo é possível que mais de uma atividade possa ser executada. Contudo, somente aquelas especificadas no *workflow* é que conduzirão o processo ao sucesso. Portanto, é necessário que existam regras que especifiquem para uma determinada situação quais ações podem ser executadas. Estas regras são denominadas regras de controle situadas (*RCS*).

As *RCS* são fórmulas do tipo $I \rightarrow E$, no qual I é o antecedente e E o conseqüente. O antecedente de uma *RCS* é uma fórmula bem formada que pode ser avaliada dentro de uma situação. O conseqüente da regra é um conjunto de atividades, e uma vez que a regra esteja satisfeita todas as atividades desse conjunto estão habilitadas para execução. De forma geral o antecedente de uma regra contém as condições para que as atividades no conseqüente possam ser executadas.

Para cada passo da projeção uma *RCS* pode ser obtida. A figura 4 apresenta uma *RCS* retirada do *workflow* apresentado na tabela 1. As regras são constituídas de conjunções ou disjunções de fatos no antecedente e atividades no conseqüente, nesse exemplo o antecedente é a condição *a_order* e o conseqüente a atividade *Montar Mercadoria*.

A síntese de regras de controle é feita com base no modelo de *workflow* gerado pelo algoritmo de planejamento. A tabela 2 apresenta o conjunto de regras geradas para o processo da tabela 1. O modelo de *workflow* gerado pelo algoritmo de planejamento *Metaplan* para a tabela 1 é dado por *M*={*(Pedido*,R. Estoque), (Pedido*, R. F. Favorito), (R. Estoque, M. Mercadoria), (R. F. Favorito, M. Mercadoria), (M. Mercadoria, C. Mercadoria), (M. Mercadoria, E. Mercadoria)*}. O símbolo * em *M* indica que a atividade possui efeito condicional. Observe que o modelo gerado é representado por um conjunto formado por pares de atividades. Esta representação é equivalente a aquela mostrada graficamente na Figura 1.

Tabela 2. Regras de controle situadas para o processo de *workflow* da tabela 1.

ID	Antecedente	Consequente
01	n_order	{Pedido}
02	ret_i	{R. Estoque}
03	ret_s	{R. Fornecedor Favorito}
04	a_order	{Montar Mercadoria}
05	Cont $\wedge$ assemb	{C. Mercadoria, Ex. Mercadoria}

O algoritmo recebe como entrada o modelo de *workflow M* no formato de pares de atividades geradas pelo *metaplan* e retorna um conjunto de regras *R*. Cada par de atividades representa uma transição do *workflow* e para cada um deles inicialmente é criado uma regra, por exemplo, para *M*(*1*)=(*Pedido,Retirar do Estoque*), o efeito da atividade *Pedido* que possui vínculo com a atividade *Retirar do Estoque* é colocado no antecedente da regra e a atividade *Retirar do Estoque* é colocado no conseqüente da regra. Assim temos que R_1=[n_*order* $\rightarrow$ {*Pedido*}].

De posse de todas as regras é preciso construir as regras que finalizam os fluxos paralelos. Para isso é preciso identificar as regras que possuem conseqüentes iguais, pois isso indica que existem dois ou mais caminhos chegando a uma mesma atividade. Quando essas regras são encontradas uma nova regra é criada com a união dos seus antecedentes e em seguida a antiga regra é excluída. Se a atividade do conseqüente for condicional, as condições serão colocadas em forma de disjunções e nesse caso qualquer uma das condições poderá acionar a regra. Caso contrário, a atividade não for condicional as condições serão colocadas em forma de conjunções. O último passo é construir as regras que iniciam um fluxo paralelo. Quando os antecedentes de uma regra são provenientes de uma mesma atividade não condicional, então os antecedentes e conseqüentes das regras são unidas em uma única regra.

A síntese de regras acontece no momento em que um modelo de *workflow* é carregado na ferramenta de execução ou durante o replanejamento.

4.3 Mecanismo de Execução

Numa rede de planos um *operador* está associado a uma situação que permita sua execução. O mecanismo de execução age com base nesta propriedade. Antes de executar uma determinada atividade associada a uma situação, o sistema verifica a existência de regras de controle acopladas a essa situação. Se existir uma regra com suas pré-condições satisfeitas ele executará as atividades especificadas pela regra. Porém, se não existir uma regra para especificar qual atividade executar o sistema não executará qualquer atividade até que alguma regra esteja associada a essa situação. Uma situação indesejada pode acontecer na falha de execução de uma atividade, ou quando eventos externos ao *workflow* interferem no andamento do processo. Quando uma falha ocorre, é possível que o estado de *workflow* caia em situações conhecidas, e para esses casos existirão regras para retornar o fluxo de execução do *workflow* para a meta. No entanto, quando ocorre de não existir regras associadas a uma situação, o mecanismo de execução recolhe as informações sobre o estado atual do *workflow* e as transforma em um problema de planejamento, no qual todas as atividades associadas ao *workflow* serão as ações, o estado atual do *workflow* será o estado inicial e a meta é mantida do modelo em execução. Observe que como é considerada a situação atual do *workflow*, o replanejamento é um processo local. Estas informações são enviadas para o planejador *Metaplan* para a criação de um novo modelo de *workflow*. O novo modelo de *workflow* gerado é transformado em novas *RCS* por meio do algoritmo de extração de regras, e assim, o conjunto de regras é atualizado com a união dos conjuntos de RCS.

Além das exceções, segundo [9], a constante adição de ramos condicionais em um *workflow* prejudica a legibilidade e a manutenção do *workflow*. Por isso, projetistas de *workflow* podem optar por modelos simplificados com o objetivo de melhorar o desempenho em relação a tempo de instanciação e recursos utilizados. Diversos w*orkflows* em grandes empresas são instanciados a todo o momento, e a redução dos recursos utilizados pode melhorar o desempenho e aumentar a capacidade de instâncias alocadas ao mesmo tempo. Baseando-se nesse fato, acrescentamos uma biblioteca de atividades associada a cada modelo de *workflow*. Uma atividade pode ser especificada durante a modelagem sem que suas pré-condições e efeitos estejam vinculados ao processo. Neste caso, durante a geração do modelo, por meio do planejador, essa atividade não fará parte do modelo de *workflow* que será instanciado nas execuções, mas a atividade será armazenada na biblioteca associada a este modelo. Durante a execução de um *workflow* atividades podem ser definidas e acrescentadas na biblioteca associada a ele sem alterar a instância em execução. As atividades em uma biblioteca só passarão a fazer parte da instância em execução após a geração de um plano válido e a criação de regras que utilizem as atividades.

Cada instância de *workflow* pode ser identificada univocamente por um número de série, que também pode ser utilizado para identificar o modelo de *workflow* que gerou a instância. O algoritmo de execução recebe como entrada uma instância de workflow e no final retorna um código com o resultado da execução. O algoritmo 1 apresenta o algoritmo de execução, no qual a função *getAtividades* retorna o conjunto de atividades do modelo de *workflow* e as atividades da biblioteca associadas ao modelo. A função *startExec* é a execução real de um conjunto de atividades que pode envolver recursos humanos e não-humanos. A variável *S* armazena a situação atual do mundo.

As funções *setRegras* e *getRegras*, configuram e retornam o conjunto de regras da instância. As funções *antecedente* e *conseqüente* retornam respectivamente a partir de uma regra o conjunto de condições no antecedente da regra e o conjunto de atividades no conseqüente da regra. A função *efeitos* retorna o conjunto de efeitos a partir de um conjunto de atividades e a função *sinteseRegras* retorna um conjunto de regras dado um modelo de *workflow* gerado pelo *metaplan*.

Segue abaixo o algoritmo de execução de *workflows*.

```
Exec(IW: InstanciaWorkflow) {
 S = {}; j = 0;
 R = IW.getRegras();
 while not S ⊇ IW.meta do
   for i = 0 to tamanho(R) do
     if S ⊇ antecedente(R(i))
       j = 1; break;
   end for;
   if j == 1
     startExec(consequente(R(i)));
     S = S – antecedente(R(i)) + efeitos(consequente(R(i)));
   else
     A = getAtividades(getModelo(IW.ID));
     M = metaplan(A,S,IW.meta);
     if M != {}
       R' = sinteseRegras(M);
       IW.setRegras(IW.getRegras() ⊇ R');
     else
       return ERROR_COD01;
   j = 0;
 end while
return EXIT_OK }
```

Como exemplo do uso do replanejamento vamos usar a tabela 1 como modelo do processo e vamos considerar que exista a atividade *Retirar de Outro Fornecedor* na biblioteca de atividades associada a esse modelo de *workflow*. A atividade *Retirar de Outro Fornecedor* possui como pré-condição *ret_o* e como efeito *a_order*., Suponha que para a fabricação de uma mercadoria, as peças necessárias não estejam disponíveis no estoque ou no fornecedor favorito que são respectivamente as atividades *Retirar do Estoque* e *Retirar do Estoque Favorito*. Isso faz com que a atividade *Pedido* se comporte de maneira inesperada, ou seja, o produto final da atividade é alterado por uma parte do sistema, ou mesmo manualmente por um gerente de projeto. Conseqüentemente não haverá regras associadas ao novo estado do *workflow*. A figura 2 ilustra esse exemplo. Na figura 2a com o surgimento do estado *ret_o* nenhuma regra poderia ser aplicada, e neste momento o replanejamento é chamado a partir dessa situação. A figura 2b apresenta o novo modelo encontrado e a partir deste modelo, novas regras são geradas e o executor poderá continuar a execução.

O reparo na instância em execução não altera as demais instâncias em execução e nem atualiza o modelo de *workflow* para as futuras execuções.

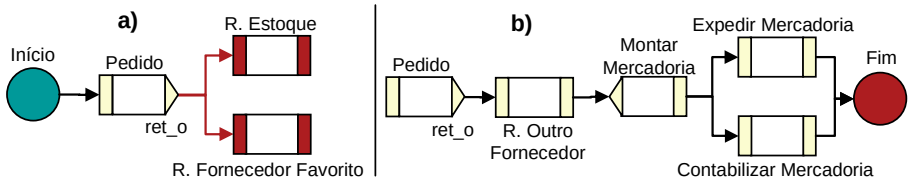

Fig. 2. Exemplo de adaptação através do replanejamento.

5 Trabalhos Relacionados

Em [10], os autores apresentam uma proposta para detecção e tratamento de exceções utilizando uma base de conhecimento. Como resultado de análise de *workflows*, desenvolveram uma crescente taxonomia de tipos de exceções. A essência do trabalho é desenvolver essa base de conhecimento e capturar tais exceções, relacioná-las com os processos que podem causá-las e com os processos que podem tratá-las. A detecção da exceção ocorre por meio da comparação manual do modelo do processo com um *template* genérico do processo, onde estão incluídos os possíveis modos de exceção. Essa abordagem exige uma base de conhecimento com vasta informação sobre o processo e sobre as formas de tratamento. Em nosso trabalho, utilizando planejamento como mecanismo de inferência, o processo de adaptação a exceções é automatizado desde que exista uma solução. Além disto, com o uso das regras de controle, as exceções são detectadas automaticamente.

O sistema gerenciador de *workflow AGENT WORK* [9], oferece um suporte para adaptação dinâmica de workflow baseado em regras. A detecção das falhas ocorre através de regras *ECA* (*event condition action*). Por meio dessas regras também são determinadas as adaptações desejadas. O sistema possui ações de controle que compõem as regras *ECAs*, permitindo remover e adicionar novas atividades, e alterar o fluxo das atividades. Esse sistema não necessita de uma grande variedade de modelos para um cruzamento de informações como em [10], e caso haja regras responde rapidamente a uma situação de exceção. No entanto, tais regras devem ser definidas para cada situação antecipadamente, ou seja, uma regra deve especificar cada atividade que será alterada. Em nossa proposta mostramos que o uso automático de planejamento, a partir de uma nova situação gerada por uma exceção, permite obter uma recomposição do fluxo de atividades automaticamente.

6 Conclusão

A utilização de sistemas gerenciadores de *workflow* com sucesso em domínios dinâmicos está condicionada a capacidade de adaptação do fluxo de *workflow* em resposta ao surgimento de exceções. No decorrer do artigo apresentamos um sistema

adaptativo de execução de *workflows* integrando planejamento com um mecanismo de execução baseado na rede de planos, e guiado por regras de controle situadas.

Por meio deste mecanismo de execução mostramos que é possível tanto detectar a exceção como oferecer uma solução para a exceção encontrada. A partir de um modelo de *workflow* gerado pelo *metaplan*, as *RCSs* podem ser extraídas e utilizadas para guiar a execução do *workflow*. Além disso, para casos onde as falhas enviem o *workflow* para estados conhecidos, as *RCS* podem ser executadas normalmente retomando o fluxo para a meta. Para casos no qual novas situações ocorram, um problema de planejamento é criado e com auxílio de uma biblioteca de atividades, novos fluxos podem ser encontrados e novas regras podem ser geradas para contemplar as necessidades do workflow. Com a combinação dessas técnicas temos um mecanismo de execução capaz de atuar em diferentes ambientes dinâmicos.

7 Referências Bibliográficas

1. Cardoso, J., Sheth, A.: Adaptation and Workflow Management Systems. In: International Conference WWW/Internet 2005, Lisboa, Portugal, pp.356-364, ISBN: 972-8924-02-X, (2005).
2. Russell, S. e Norving, P.: Artificial Intelligence – A Modern Approach, 2ª ed,. Prentice Hall, (2003).
3. PLANET Workflow Management R&D RoadMap, www.planet-noe.org.
4. Drummond, M.: Situated control rules. In: First International Conference on Principles of Knowledge Representation and Reasoning, pp. 103-113. Toronto, (1989).
5. Melo, J., T., Silveira, L., B., Lopes, C., R.: Modelando e Executando Workflow com técnicas de planejamento apoiado em Inteligência Artificial. In: International Conference on Information Sysems and Technology Management, São Paulo, (2006).
6. WFMC - Workflow Management Coalition: The workflow reference model, http://www.wfmc.org, (2004).
7. Fikes, R. E. & Nilsson, N. J.: STRIPS: A new approach to the application of theorem proving to problem solving. Artificial Intelligence 2, 189–208, (1971).
8. Hoffmann J., Nebel B.: The FF Planning System: Fast Plan Generation Through Heuristic Search. In: Journal of Artificial Intelligence Research, Volume 14, Pages 253 – 302, (2001).
9. Muller, R.; Greiner, U.; Rahm, E.: AGENT WORK: A workflow system supporting rule-based workflow adaptation. Data & Knowledge Engineering. Amsterdam, vol. 51, Issue 2, pp. 223-256, Editora: Elsevier Science Publishers B. V., ISSN: 0169-023X, (2004).
10. Klein, M., Dellarocas, C.: A Knowledge-Based Approach to Handling Exceptions in Workflow Systems. In: Journal of Computer Supported Collaborative Work, (2000).