Um sistema gerenciador de *Workflow* de acordo com o método *Catalysis*

Edson Alves de Oliveira Junior e Itana Maria de Souza Gimenes*

Departamento de Informática, Universidade Estadual de Maringá, Av. Colombo, 5790, 87020-900, Maringá-Paraná, Brasil.
**Autor para correspondência. e-mail: itana@din.uem.br*

RESUMO. Este artigo apresenta a especificação de um sistema gerenciador de workflow de acordo com a abordagem de desenvolvimento baseado em componentes, seguindo o método Catalysis. A especificação é resultado de pesquisas iniciadas no contexto do ambiente ExPSEE, específico para processos de software, ampliadas para abranger características de gerenciadores de workflow. Com base na arquitetura e nos componentes propostos, pode-se projetar gerenciadores de workflow de propósito geral ou específico. Assim, os resultados apresentados neste trabalho contribuem para apresentar uma evolução dos modelos e das arquiteturas propostas pela WfMC, em UML, mostrar como o processo de desenvolvimento do Catalysis foi modelado em uma ferramenta comercial (Rational Rose) e apresentar componentes reutilizáveis ao contexto dos gerenciadores de workflow.

Palavras-chave: catalysis, desenvolvimento baseado em componentes, rational rose, reuso, UML, sistemas de gerenciamento de workflow.

ABSTRACT. A Workflow management system according to the Catalysis method. This paper presents a workflow management system specification according to a component-based development approach, following the Catalysis method. The specification results from research projects within the context of the ExPSEE environment. This environment, specific for software processes, was extended to enclose features of workflow managers. Based on the architecture and on the proposed components, workflow managers of general or specific purposes can be designed. Thus, the results presented in this paper contribute to present an evolution of the models and architectures proposed by WfMC, in UML; to show how the Catalysis development process was designed in a commercial tool (Rational Rose); and to present reusable components to the context of workflow managers.

Key words: catalysis, component-based development, rational rose, reuse, UML, workflow management systems.

Introdução

O ambiente ExPSEE (*Experimental Process-centred Software Engineering Environment*) vem sendo desenvolvido desde 1994 pelo grupo de engenharia de software da UEM e foi concebido como um ambiente de engenharia de software orientado a processos. Esse ambiente tem como base de implementação a plataforma *Sun Solaris* e utiliza alguns mecanismos proprietários para realizar as suas funcionalidades. O contexto desse ambiente é especificamente o de processo de software, apoiado por ferramentas CASE e mecanismos de modelagem e automação.

Com o crescente aumento do uso dos conceitos de *workflow*, foi possível propor uma arquitetura de componentes de software para o ambiente ExPSEE, compatível com a arquitetura genérica proposta pela *Workflow Management Coalition* (WfMC) (WfMC, 1995). Essa arquitetura é composta por componentes genéricos a qualquer sistema gerenciador de *workflow*, geral ou específico.

A abordagem de Desenvolvimento Baseado em Componentes (DBC) vem sendo objeto de estudo nos últimos anos ganhando cada vez mais espaço tanto na comunidade acadêmica quanto empresarial. O DBC permite a criação de componentes que podem ser reutilizados em outros projetos e/ou, ainda, a agregação de componentes prontos (Componentes COTS[1]) à aplicação que está sendo desenvolvida.

[1] Commercial Off-The-Shelf

De acordo com esses conceitos, este artigo apresenta a especificação de um sistema gerenciador de *workflow*, baseando-se nas técnicas de DBC, mais especificamente no método *Catalysis*, tomando a *Unified Modelling Language* (UML) (OMG, 2002) como notação apoiada pela ferramenta CASE[2] Rational Rose (Rational Software, 2002). A partir dessa especificação, é possível projetar qualquer tipo de gerenciador de *workflow*, geral ou específico, tomando como referência os componentes genéricos, de acordo com a WfMC, que formam a arquitetura do sistema aqui proposto.

Assim, as seções seguintes apresentam: as versões do ambiente ExPSEE; a especificação do sistema gerenciador de *workflow* proposto, com base na arquitetura de componentes concebida para o ambiente ExPSEE, e as conclusões e contribuições deste trabalho.

O Ambiente ExPSEE

O ExPSEE é um ambiente que oferece mecanismos de apoio à modelagem e à automação de processos, tendo em vista a utilização de qualquer ciclo de vida que envolva ferramentas de suporte às suas diversas fases. Esse ambiente baseia-se na definição explícita do processo por meio do qual os artefatos serão concebidos, projetados, desenvolvidos e distribuídos. Essa definição explícita do processo baseia-se no modelo de processo adotado para o ambiente.

A seguir, são apresentadas: a versão operacional e os mecanismos utilizados para a sua construção; a arquitetura de componentes proposta para o ambiente e a variante *Open Source* e os mecanismos definidos para a migração do ambiente ExPSEE da plataforma *Sun Solaris* para a plataforma Linux.

A versão operacional do ambiente ExPSEE

A construção de um ambiente de engenharia de software requer a definição de uma série de mecanismos que permitam a execução das funcionalidades disponíveis no ambiente. Esses mecanismos podem ser divididos em quatro categorias, a saber: linguagem de programação; construtores de interface com o usuário; padrões de comunicação interprocessos e Sistemas Gerenciadores de Banco de Dados (SGBD).

O ExPSEE possui três subsistemas principais que são responsáveis pelo gerenciamento de processos, dos dados e da interface com o usuário.

O **gerenciamento dos dados** corresponde aos serviços desempenhados pelo *ObjectStore* (Gimenes, 1999). **O gerenciamento da interface** com o usuário contém um conjunto de interfaces utilizado pelo gerenciador de processos para fazer a comunicação com o usuário. O **gerenciador de processos** do ExPSEE baseia-se no padrão *Process Manager* (Weiss, 1998).

O gerenciador de processos é o subsistema mais importante do ambiente ExPSEE, pois é responsável por controlar a criação, a modificação e a execução dos processos de software, dando manutenção ao estado atual de cada tarefa e do processo como um todo. O gerenciador de processos também controla a alocação de recursos utilizados nas tarefas, bem como os cargos do processo, assegurando que nenhum direito seja violado (Gimenes, 1999).

A versão operacional do ambiente ExPSEE foi implementada sobre a plataforma *Sun Solaris*. As linguagens de programação utilizadas para a implementação dos módulos gerenciadores e de interface foram, respectivamente, C++ e Tcl/Tk.

Para a criação e a manipulação dos objetos necessários para o escalonamento das tarefas, foi utilizado *ObjectStore*, enquanto que RPC (*Remote Procedure Call*) foi utilizado como mecanismo de comunicação interprocessos.

A arquitetura de componentes proposta para o ambiente ExPSEE

Com o emergente crescimento e com a disseminação dos conceitos de *workflow*, foi possível conceber para o ambiente ExPSEE uma arquitetura baseada em componentes reutilizáveis compatível com a arquitetura genérica da *Workflow Management Coalition* (WfMC, 1995) (Figura 1).

A arquitetura de componentes proposta tomou como base o padrão *Process Manager* e serve para orientar o projeto interno e a integração dos componentes do ambiente ExPSEE. Essa arquitetura contém informações importantes sobre o domínio, tais como as características dos relacionamentos entre os componentes e, ainda, a informações sobre a integração do sistema como um todo.

Essa arquitetura é formada pelos seguintes componentes (Lazilha, 2002):

***GraphicalInterfaceMgr*,** responsável pela interação do usuário com o sistema gerenciador de processos;

***WorkflowArchitectureMgr*,** responsável pelo controle e pelo gerenciamento da construção e da manutenção de arquiteturas de processo de *workflow*;

***WorkflowMgr*,** responsável pela criação e gerenciamento de projetos, através da instanciação e execução dos seus processos;

[2] Computer-Aided Software Engineering

***TaskScheduler*,** responsável pelo controle e gerenciamento das tarefas e das ações a serem realizadas no processo de *workflow*;

***ResourceAllocationMgr*,** responsável pela alocação de recursos necessários como, por exemplo, atores, ferramentas e artefatos para a realização dos projetos;

***ExternalApplicationMgr*,** responsável pelo gerenciamento das aplicações externas invocadas durante a definição e a execução de um processo de *workflow*;

***ObjectMgr*,** responsável pelo relacionamento com o sistema gerenciador de banco de dados;

***Interpreter*,** permite à linguagem de programação de processos executar as tarefas através do gerenciador de tarefas; e

WorkflowExecutionMgr, responsável pelo controle e gerenciamento das tarefas a serem realizadas no *workflow*, sendo que a sua principal característica é apoiar a interação com os usuários do WfMS[3], auxiliando-os na identificação de suas tarefas no *workflow*.

Uma descrição mais detalhada de cada um dos componentes e suas respectivas interfaces e métodos serão apresentadas na seção seguinte.

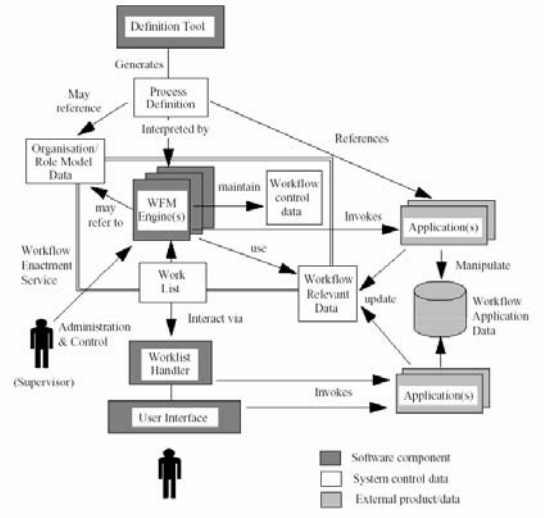

Figura 1. Arquitetura genérica de um sistema de *Workflow* (WfMC, 1995).

A Variante *Open Source* do ambiente ExPSEE

O conceito *Open Source* surgiu da proposta de se padronizar a produção de sistemas de computador que podem ser livremente distribuídos, modificados e redistribuídos. Dentre as vantagens de seguir esse conceito, está a isenção de taxas ou "royalty" para adquirir produtos conformantes a esse conceito.

Baseando-se nos conceitos de *Open Source* e DBC, foi possível propor uma variante para a versão atual do ambiente ExPSEE (Oliveira Junior, 2003).

Essa variante do ExPSEE utiliza somente recursos *Open Source*. Como base é utilizada a plataforma Linux, além dos seguintes mecanismos (Oliveira Junior, 2002): **Linguagem de Programação**: Java, por fazer parte de uma tecnologia que dispõe de uma série de recursos para utilização em aplicações tanto científicas quanto comerciais; **Construção de Interface com o Usuário (GUI[4])**: o *toolkit* Swing (Java 2), devido à sua constante atualização e a uma ampla coleção de elementos gráficos de que dispõe, e o *framework* JHotDraw, por possuir uma API bastante simples que dispõe de vários recursos para a manipulação de objetos gráficos em aplicações Java; **Padrão de Comunicação Interprocessos**: CORBA, mais especificamente o produto CORBA JacORB, que é um ORB (*Object Request Broker*) implementado em Java e possui várias funcionalidades não identificadas em outros produtos; **Sistema Gerenciador de Banco de Dados**: o SGBD relacional MySQL, como mecanismo de armazenamento persistente, e o *framework* ObjectBridge, por ser escrito em Java e permitir o mapeamento de objetos Java para o MySQL através de XML (*Extensible Markup Language*).

Especificação do Sistema Gerenciador de *Workflow*

Apesar do ExPSEE possuir uma arquitetura compatível com a arquitetura da WfMC, organizada em componentes reutilizáveis, é necessária uma especificação que permita o gerenciamento e a implementação de componentes seguindo um padrão.

A especificação do ExPSEE baseia-se na abordagem de DBC e tem como notação a linguagem UML (OMG, 2002) apoiada pelo método *Catalysis* (D'Souza, 1999) e a ferramenta Rational Rose (Rational Software, 2002). O método *Catalysis* foi adotado devido a sua forte ênfase na produção de *frameworks* e de componentes.

O método *Catalysis* baseia-se na linguagem de modelagem UML e alinha-se aos padrões de sistemas de objetos distribuídos abertos, construídos a partir de componentes e *frameworks*. *Catalysis* oferece modelos que permitem aos desenvolvedores partirem da análise e especificação do domínio da

[3] Workflow Management System

[4] Graphical User Interface

aplicação e chegarem ao código, dividindo o sistema e identificando ao longo do processo os elementos de reutilização.

Existe uma grande preocupação em relação à especificação dos componentes e de suas interfaces. Por isso, existem recursos, como a linguagem OCL (*Object Constraint Language*) (OMG, 2002), que permite formalizar a definição das interfaces e dos componentes especificando restrições e detalhes de forma que possam ser verificados formalmente.

Foram definidos quatro estágios para o *Catalysis*, baseados nas fases de desenvolvimento de sistemas de informação (D'Souza, 1999), sendo eles: **Especificação de Requisitos (*Requirements*):** visa entender e representar o problema definindo o contexto do sistema envolvido e produzindo um modelo do domínio; **Especificação do Sistema (*System Specification*):** trata a solução de software extraída do modelo do domínio, representando-se os tipos e as operações do sistema e seu comportamento exterior; **Projeto da Arquitetura (*Architectural Design*):** visto em duas partes: arquitetura da aplicação - representa a estrutura lógica do sistema como uma coleção de componentes cooperantes com os tipos e operações obtidos na especificação, distribuídos através dos componentes e arquitetura técnica - inclui as partes do sistema independentes do domínio como a infra-estrutura de comunicação de componentes, a plataforma de hardware e a plataforma de software e **Projeto Interno dos Componentes (*Component Internal Design*):** envolve a definição das classes e interfaces dos componentes, a construção (código) e o teste dos componentes.

A ferramenta Rational Rose baseia-se no Processo Unificado da Rational Software (*Rational Unified Process - RUP*) (Rational Software, 2002). Porém, o método *Catalysis* não possui estágios compatíveis com o RUP e, dessa forma, foi definido um *framework* de processo para o *Catalysis* na Rational Rose (Figura 2). Os quatro estágios do *Catalysis* foram distribuídos entre os três pacotes inerentes à Rational Rose, *use case view*, *logical view* e *component view*, definindo, então, a estrutura do *framework* da seguinte maneira (Oliveira Junior, 2003): ***use case view***: *requirements;* ***logical view***: *system specification* e *architectural design*; e ***component view***: *component internal design*.

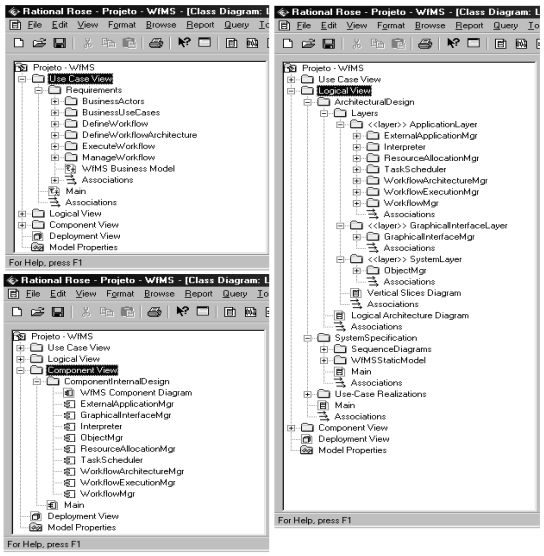

Figura 2. *Framework* de processo para o método *Catalysis* (Oliveira Junior, 2003).

A seguir, é apresentada a especificação do sistema gerenciador de *workflow*. A especificação completa encontra-se em (Oliveira Junior, 2003).

Especificação de requisitos

Neste estágio é identificado o domínio da aplicação representando objetos e ações do domínio em questão, no caso, sistemas de gerenciamento de *workflow*.

Vários diagramas foram gerados para especificar o domínio da aplicação, sendo: um diagrama de caso de uso que representa o modelo de negócio; quatro diagramas de caso de uso representando as expansões dos casos de uso identificados no modelo de negócio e dois diagramas de classe representando os modelos estáticos de tipos do domínio da aplicação. Assim, são apresentados o modelo de negócios (Figura 3), a definição de um *workflow* (Figura 4) e o modelo de tipos da definição de um *workflow* (Figura 5).

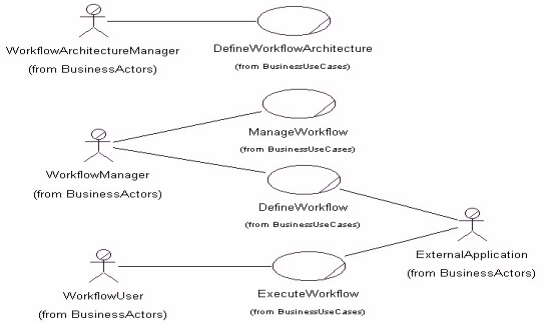

Figura 3. Modelo de Negócios de um WfMS (Oliveira Junior, 2003).

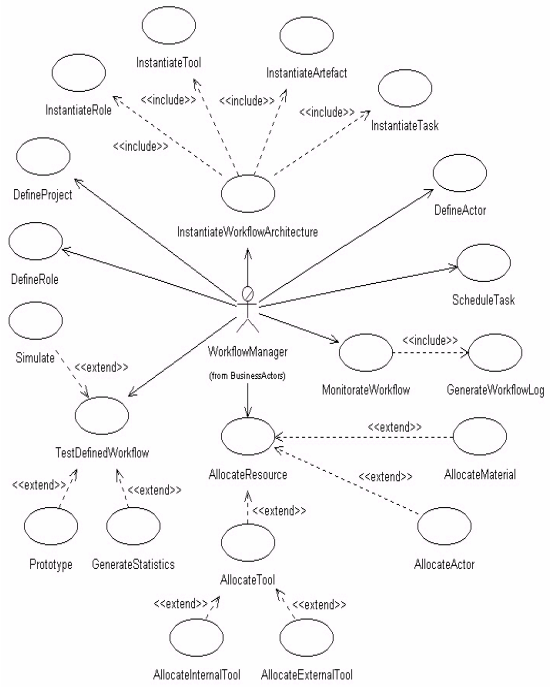

Figura 4. Diagrama de Casos de Uso: *Define Workflow* (Oliveira Junior, 2003).

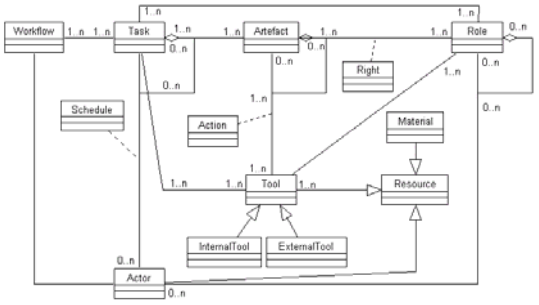

Figura 5. Modelo Estático de Tipos: *Define Workflow* (Oliveira Junior, 2003).

Especificação do sistema

Neste estágio é tratada a modelagem da solução de software identificada nos modelos de domínio obtidos na fase de especificação de requisitos. A análise das ações do sistema representadas nos diagramas de caso de uso torna possível a identificação dos tipos, permitindo a associação das referidas ações aos respectivos tipos relacionados.

Assim, são apresentados o modelo estático de tipos de um WfMS (Figura 6) e o diagrama de seqüência do caso de uso *define workflow* (Figura 7).

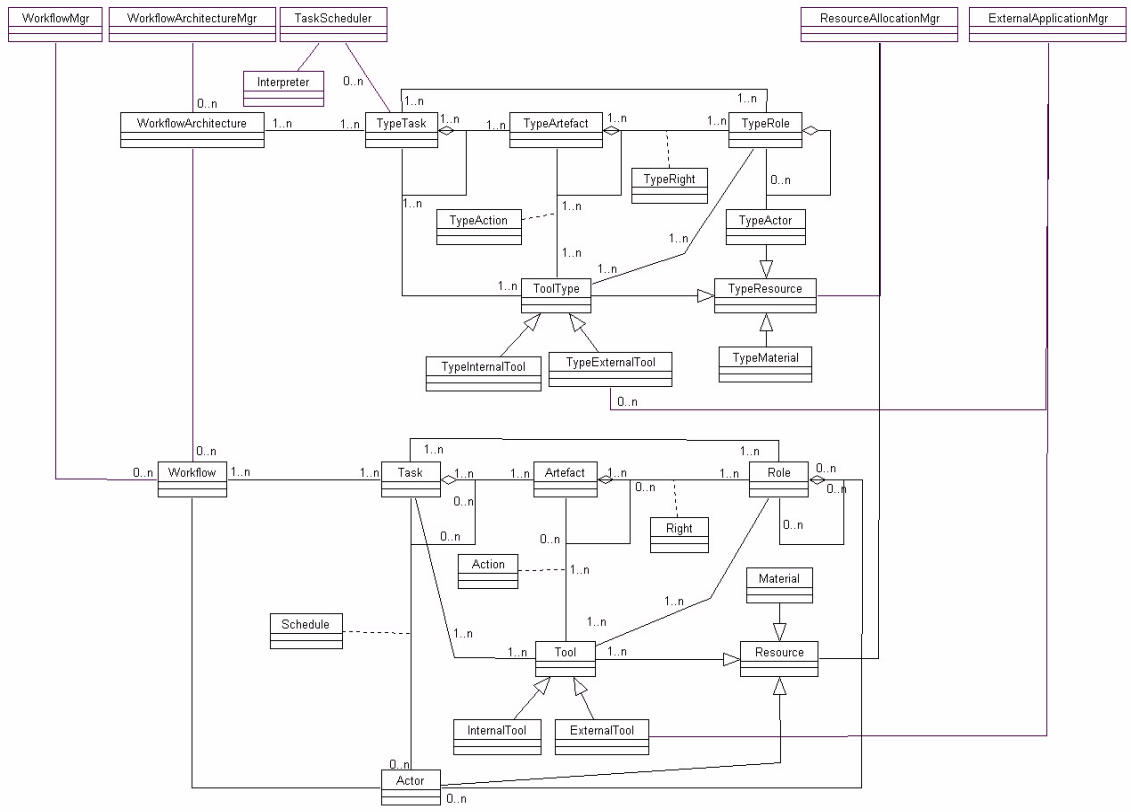

Figura 6. Modelo Estático de Tipos de um WfMS (Oliveira Junior, 2003).

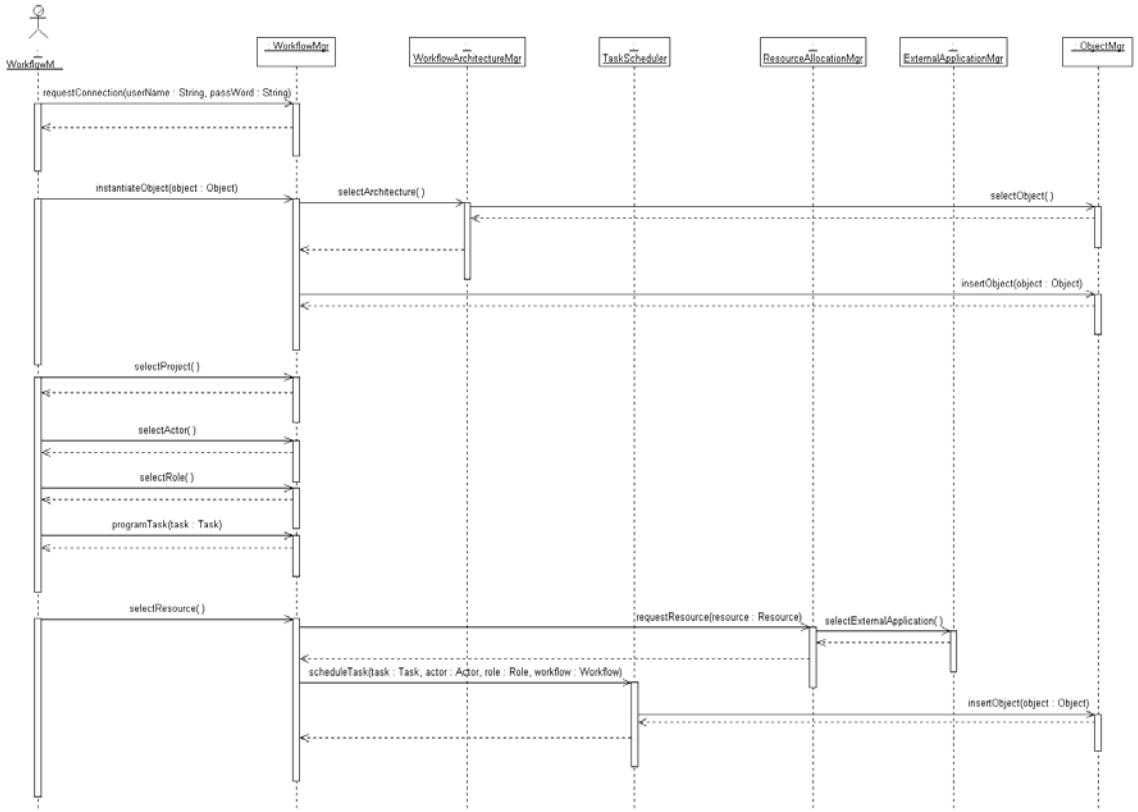

Figura 7. Diagrama de Seqüência: *Define Workflow* (Oliveira Junior, 2003).

Projeto da arquitetura

O projeto da arquitetura de um sistema de software visa definir suas estruturas gerais, descrevendo os elementos que compõem os sistemas e as interações entre estes apoiando também questões importantes de projeto, como, por exemplo, a organização do sistema como composição de componentes.

Assim, a Figura 8 apresenta a arquitetura lógica de um WfMS.

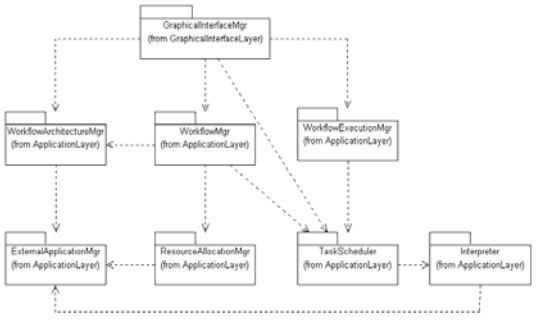

Figura 8. Arquitetura Lógica de um WfMS (Oliveira Junior, 2003).

Projeto Interno dos Componentes

Neste estágio os componentes individuais da aplicação são refinados até o nível de interfaces, classes ou componentes pré-existentes em uma linguagem de programação. Para tal, a implementação deve satisfazer os requisitos funcionais e não-funcionais definidos na especificação do sistema, assim como seguir a arquitetura definida.

A Figura 9 apresenta a arquitetura de componentes para um WfMS.

A seguir, é apresentada uma descrição da funcionalidade de cada um dos componentes da arquitetura proposta do sistema gerenciador de *workflow,* bem como as suas respectivas interfaces e métodos (Oliveira Junior, 2003) e seus aspectos de variabilidade (Lazilha, 2002):

GraphicalInterfaceMgr: esse componente é responsável pelo gerenciamento da interface com o usuário do sistema. O ponto de variação desse componente está no fato de que a interação com o usuário pode ocorrer via interface gráfica convencional ou via *browser*.

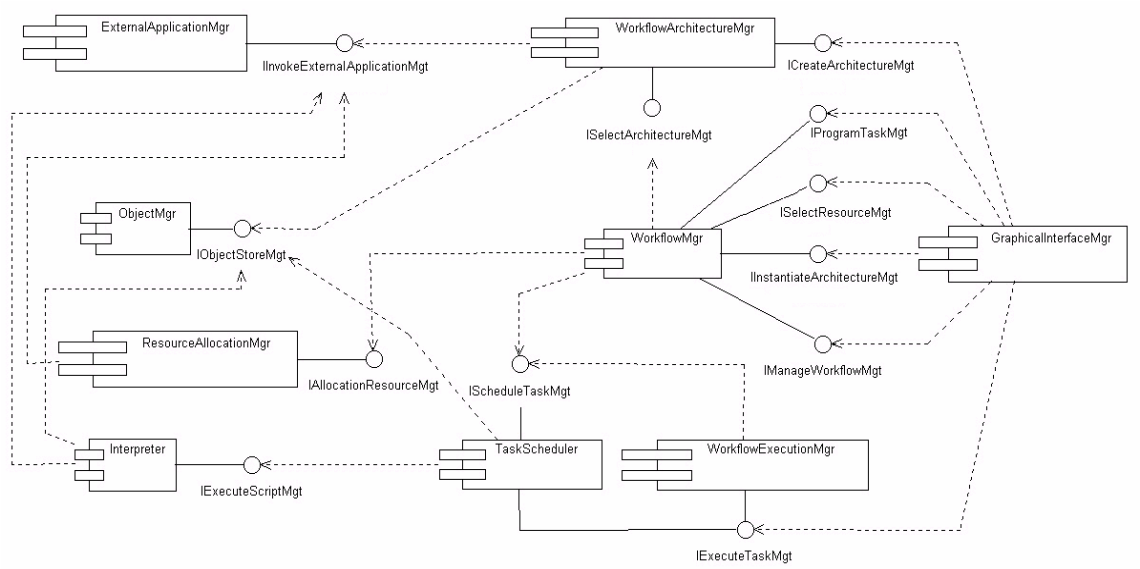

Figura 9. Arquitetura de Componentes de um WfMS (Oliveira Junior, 2003)

WorkflowArchitectureMgr: esse componente é responsável pelo controle e gerenciamento da construção e da manutenção de arquiteturas de *workflow*. A implementação desse componente suporta as funções relacionadas à definição das arquiteturas de *workflow* e os tipos de objetos relacionados a esta. A utilização desse componente torna a definição de *workflows* mais flexível, visto que, dessa forma, os tipos de objetos não são pré-fixados. Diferentes linguagens de programação de processos podem ser apoiadas para a definição das tarefas, que posteriormente serão executadas pelo interpretador.

ICreateArchitectureMgt

requestConnection (String userName, String passWord): boolean	Permite a conexão de um usuário.
createObject (Object object): boolean	Cria um novo objeto de definição de *workflow* (ex. um ator, cargo, uma ferramenta).
deleteObject (Object object): boolean	Exclui um objeto de definição de *workflow*.
retrievalObject (Object object): boolean	Recupera um objeto de definição de *workflow*.
insertObjectAttribute (Object objectAttribute): boolean	Insere um atributo específico em um objeto de definição.
deleteObjectAttribute (Object objectAttribute): boolean	Exclui um atributo específico em um objeto de definição.
setBaselineArchitecture (WorkflowArchitecture architecture): boolean	Salva uma arquitetura de referência.

ISelectArchitectureMgt

selectArchitecture (WorkflowArchitecture architecture): WorkflowArchitecture	Seleciona uma arquitetura de *workflow*.

WorkflowMgr: esse componente é responsável pela criação e gerenciamento de projetos que incorporam *workflows*. Os projetos incluem a instanciação e execução de arquiteturas de *workflow*. A cada projeto existe um *workflow* associado. Para cada tipo de objeto existente na arquitetura, instancia-se um objeto no *workflow*. Em seguida, as tarefas do projeto são executadas e gerenciadas, fazendo-se a alocação de recursos e demais tomadas de decisão.

IInstantiateArchitectureMgt

requestConnection (String userName, String passWord): boolean	Permite a conexão de um usuário.
selectArchitecture (WorkflowArchitecture architecture): WorkflowArchitecture	Seleciona uma arquitetura de *workflow* para instanciação.
instantiateObject (Object object): boolean	Instancia objetos definidos na arquitetura de *workflow* selecionada.

IProgramTaskMgt

selectProject (Object project): Object	Seleciona projeto a ser executado.
selectTask (Task task): Task	Seleciona tarefa a ser codificada.
selectRole (Role role): Role	Seleciona cargo ao qual a tarefa a ser codificada está associada.
selectActor (Actor actor): Actor	Seleciona ator para a execução da tarefa.
programTask (Task task): boolean	Programa uma determinada tarefa.

ISelectResourceMgt

selectResource (Resource resource): Resource	Seleciona recursos que serão usados na execução de uma determinada tarefa.

IManageWorkflowMgt

selectResource (Resource resource): Resource	Seleciona recursos que serão usados na execução de uma determinada tarefa.

WorkflowExecutionMgr: esse componente é responsável pelo controle e gerenciamento das tarefas a serem realizadas no *workflow*. Sua principal característica é apoiar a interação com os usuários do WfMS. É através dele que os usuários identificam as suas tarefas no *workflow*.

IExecuteTaskMgt

requestConnection (String username, String passWord): boolean	Permite a conexão de um usuário.
selectTask(Task task): Task	Seleciona uma determinada tarefa.
executeTask(Task task) : boolean	Executa uma determinada tarefa.
visualizeTask (Task task, String newDate, Actor actor): Task[]	Mostra as tarefas pertencentes a um determinado usuário de acordo com o seu cargo.

TaskScheduler: esse componente é responsável pelo controle e gerenciamento das tarefas e ações a serem realizadas. É por meio desse componente que os usuários interagem com o sistema gerenciador de *workflow*. O *TaskScheduler* permite que os usuários identifiquem suas tarefas geradas pelo *WorkflowMgr*.

IScheduleTaskMgt

requestConnection (String userName, String passWord): boolean	Permite a conexão de um usuário.
scheduleTask (Task task, Actor actor, Role role, Workflow workflow): boolean	Agenda uma determinada tarefa.

IExecuteTaskMgt

selectTask (Task task): Task	Seleciona uma determinada tarefa.
executeTask (Task task): boolean	Executa uma determinada tarefa.
cancelTask (Task task): boolean	Cancela uma determinada tarefa a ser executada ou que já está em execução por um determinado usuário.
interruptTask (Task task) : boolean	Interrompe uma determinada tarefa.
restartTask (Task task): boolean	Reinicia uma determinada tarefa.
finalizeTask (Task task) : boolean	Finaliza uma determinada tarefa.
visualizeTask (Task task, String newDate, Actor actor) : Task[]	Mostra as tarefas pertencentes a um determinado usuário de acordo com o seu cargo.

ResourceAllocationMgr: esse componente é responsável pela alocação de recursos (atores, ferramentas ou material).

IAllocationResourceMgt

requestResource (Resource resource): boolean	Solicita um determinado recurso (ex. ferramenta, ator).
releaseResource (Resource resource): boolean	Libera um determinado recurso para efetiva utilização.
deleteResource (Resource resource): boolean	Exclui um determinado recurso.
retrievalResourceStatus (Resource resource): Resource	Recupera o estado de um determinado recurso (ex. reservado).

ExternalApplicationMgr: esse componente é responsável pelo gerenciamento das aplicações externas invocadas durante a definição do *workflow* e execução das tarefas.

IInvokeExternalApplicationMgt

selectExternalApplication (ExternalApplication externalApplication): ExternalApplication	Seleciona uma aplicação externa.
invokeExternalApplication (ExternalApplication externalApplication): boolean	Invoca uma aplicação externa.

ObjectMgr: esse componente é responsável pelo relacionamento com os mecanismos de armazenamento de objetos manipulados pelo sistema. Suas funções trabalham com objetos, que podem ser considerados desde dados de controle do processo, dados relevantes do processo ou, até mesmo, instâncias de *workflow* e tarefas. Todos os componentes da arquitetura utilizam seus serviços. Sua presença torna o restante dos componentes independentes de uma implementação particular de um sistema gerenciador de objetos, garantindo flexibilidade e portabilidade para toda a coleção de componentes existentes.

IObjectStoreMgt

insertObject (Object object): boolean	Insere um novo objeto de *workflow*.
updateObject (Object object): boolean	Atualiza um objeto de *workflow*.
deleteObject (Object object): boolean	Exclui um objeto de *workflow*.
selectObject (Object object): Object	Seleciona um objeto de *workflow*.

Interpreter: uma linguagem para programação de processos é necessária para que o processo possa ser programado e executado pelo *WorkflowMgr*. As tarefas que compõem o *workflow* são então programadas, utilizando uma linguagem de programação de processos e de recursos que permitem a execução cooperativa dessas tarefas. Para a execução dessas tarefas, o *TaskScheduler* realiza chamadas ao interpretador da linguagem para que este possa executá-las.

IExecuteScriptMgt

executeScript (String script): boolean	Executa Script.
invokeExternalApplication (ExternalApplication externalApplication): boolean	Invoca uma aplicação externa.

Neste trabalho também foi utilizada OCL (*Object Constraint Language*) por ser uma linguagem de expressão formal. A OCL não possui efeito sobre a UML, pois o estado do sistema não se modifica com a aplicação da OCL. Além disso, OCL não é uma linguagem de programação. Todas as construções da linguagem possuem um significado formal (OMG, 2002).

A Figura 10 apresenta o código OCL de alguns métodos da interface *IExecuteTaskMgt* associada ao componente *TaskScheduler*. Os demais são apresentados em Oliveira Junior (2003).

```
context IExecuteTaskMgt :: interruptTask(task : Task) : Boolean
pre: -- The task must be in state 6 (Executing).
  self.task.status = 6
post: -- The task must be in state 5 (Suspended).
  if self.task.status = 5 then result = true
  else result = false
  endif

context IExecuteTaskMgt :: visualizeTask(actor : Actor, role : Role)
: Set(Task)
pre: -- The role name must be "WORKFLOW MANAGER" or "WORKFLOW USER".
  self.role.roleName = "WORKFLOW MANAGER" or self.role.roleName =
"SOFTWARE ENGINEER"
post:  if self.role.roleName = "WORKFLOW MANAGER" then result =
Task.allInstances
         else
           if self.role.roleName = "WORKFLOW USER" then
             result = Task.allInstances -> Role.includes("WORKFLOW
USER")
           endif
         endif
```

Figura 10. Código OCL Parcial do Componente *TaskScheduler* (Oliveira Junior, 2003).

Conclusão

O processo de DBC tem contribuído com a redução dos custos de desenvolvimento de software promovendo a reutilização de componentes em vários projetos. Os sistemas gerenciadores de *workflow* estão se tornando cada vez mais presentes na automatização dos processos de negócios das empresas. A aquisição de um sistema de gerenciamento de *workflow* requer um alto investimento financeiro para as empresas. Assim, uma alternativa para essas empresas é a construção de seus próprios sistemas gerenciadores de *workflow*, de acordo com as suas necessidades específicas.

A especificação do sistema gerenciador de *workflow*, apresentada neste artigo, contribui para a geração de gerenciadores de *workflow* gerais ou específicos. Como os componentes estão sendo especificados de acordo com uma abordagem de desenvolvimento baseado em componentes, como o método *Catalysis*, será possível reutilizá-los em outros projetos ou, até mesmo, agregar novos componentes ao sistema que se deseja construir.

A especificação apresentada permite a geração de componentes que podem comunicar-se através do padrão CORBA em ambientes distribuídos, devido às suas interfaces bem definidas em IDL.

A arquitetura de componentes apresentada neste artigo está sendo usada como base para projetar uma arquitetura de linha de produto para sistemas gerenciadores de *workflow* (Lazilha, 2002). Essa arquitetura de linha de produto permitirá a geração de produtos de *workflow* de baixo custo e configuráveis de acordo com as necessidades das empresas. Trabalhos relacionados estão desenvolvendo o mecanismo de geração de produto, e alguns componentes da arquitetura, sendo que os demais componentes já podem ser reutilizados. Acreditamos que a validação da arquitetura deverá ser realizada ao término da especificação e implementação de todos os componentes de tal arquitetura, utilizando, para isso, ferramentas de simulação ADL (*Architectural Definition Language*) e prototipação.

Este artigo apresentou, também, o processo de desenvolvimento do método *Catalysis* modelado em uma ferramenta comercial de ampla utilização, como a Rational Rose, gerando um *framework* de processo para o *Catalysis*. Além disso, foram apresentados componentes reutilizáveis ao contexto dos gerenciadores de *workflow*, servindo de referência para outros trabalhos. A partir dessa especificação, pode-se efetivamente gerar código Java e fazer as adaptações necessárias para produzir variantes de gerenciadores de *workflow*.

Referências

D'SOUZA, D.F., WILLS, A.C. Objects, components, and frameworks with UML - The Catalysis Approach. Massachusetts: Addison-Wesley Publishing Company, 1999..

GIMENES, I.M.S. ExPSEE - An experimental process-centred software engineering environment. 1999. Universidade Estadual de Maringá, Maringá, 1999. (Relatório Final, PPG)

LAZILHA, F.R. *Uma Proposta de Arquitetura de Linha de Produto para Sistemas de Gerenciamento de Workflow*. 2002. Dissertação (Mestrado) – PPGC, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2002.

OLIVEIRA JUNIOR, E.A.; GIMENES, I.M.S. Portando o ExPSEE para a Plataforma Linux. 2002. Universidade Estadual de Maringá, Maringá, 2002. (Relatório Semestral de Iniciação Científica, PPG)

OLIVEIRA JUNIOR, E.A. *Especificação do ambiente ExPSEE de acordo com a abordagem de desenvolvimento baseado em componentes*. 2003. Monografia (Graduação em Informática) – Departamento de Informática, Universidade Estadual de Maringá, Maringá, 2003.

OMG - OBJECT MANAGEMENT GROUP. 2002. Disponível em <http://www.omg.org>. Acesso em: 10 dez. 2002.

RATIONAL SOFTWARE. 2002. Disponível em <http://www.rational.com>. Acesso em: 20 dez. 2002.

WEISS, G.M. *Um padrão para definição de um gerenciador de processos de software*. 1998. Monografia (Graduação em Ciência da Computação) – Departamento de Informática, Universidade Estadual de Maringá, Maringá, 1998.

WfMC - WORKFLOW MANAGEMENT COALITION. The Workflow Reference Model. 1995. Disponível em <http://www.wfmc.org>. Acesso em: 15 set. 2002.

Received on March 02, 2003.
Accepted on June 25, 2003.