Uma Proposta de Arquitetura de Linha de Produtos para Workflow Management Systems

Fabrício R. Lazilha[1,3] Itana M. S. Gimenes[2] R. T. Price[3]
[1]Centro Universitário de Maringá
[2]Universidade Estadual de Maringá
[3]Programa de Pós-Graduação em Computação – Universidade Federal do Rio Grande do Sul
E-mail: fabrício@cesumar.br / itana@din.uem.br / tomprice@inf.ufrgs.br

Resumo

Este artigo apresenta uma proposta de arquitetura de linha de produtos para sistemas de gerenciamento de workflow. O processo seguido para definição da arquitetura de linha de produtos e a notação utilizada para a representação das variabilidades são descritos. O domínio de sistemas de gerenciamento de workflow tem se mostrado altamente favorável à aplicação da abordagem de linha de produtos, pois existe uma arquitetura padrão e uma demanda por produtos similares, porém com características diferentes.
Palavras-chave: Sistema de Gerenciamento de *Workflow*, Linha de produto, Arquitetura de software, Reutilização.

Abstract

This paper proposes a software product line architecture for workflow management systems. It describes the process follow to obtain the architecture as well as the notation used to represent variabilities. The workflow management system domain has proved to be feasible to the application of the product line approach. This is because there is a standard architecture and a call for similar products but which have main different features. The architecture was validated simulating it using RAPIDE tools.
Keywords: Workflow Management Systems, Product Line, Software Architecture, Reuse.

1. Introdução

De acordo com Bass [1], a abordagem de linha de produtos de software cria uma coleção de sistemas que compartilham um conjunto gerenciado de características entre seus principais artefatos. Estes artefatos incluem uma arquitetura base e um conjunto de componentes comuns para preencher esta arquitetura. O projeto de uma arquitetura para uma família de produtos deve considerar as semelhanças e variabilidades entre os produtos desta família.

A abordagem de linha de produtos é aplicável aos domínios em que existe uma demanda por produtos específicos, no entanto existe um conjunto de características comuns e pontos de variabilidade bem definidos. O domínio dos Sistemas de Gerenciamento de *Workflow* (WfMS) é altamente favorável à aplicação da abordagem de linha de produtos, devido aos esforços da *Workflow Management Coalition* (WfMC) [2], que viabilizou a construção de uma arquitetura genérica para WfMS que pode ser ajustada à maioria dos produtos do mercado. A partir disso a WfMC estabeleceu um modelo de referência para WfMS. Cada implementação de um WfMS pode adaptar componentes ou interfaces de acordo com as necessidades da aplicação. Produtos de *workflow* com características similares, porém com diferentes especificidades são necessários nas mais diversas empresas que utilizam esta tecnologia.

Este artigo apresenta o processo seguido para o projeto de uma arquitetura de linha de produtos para WfMS. Apresenta também as extensões utilizadas para representar variabilidade no processo de desenvolvimento. A seção 2 descreve o processo de desenvolvimento da arquitetura proposta e as extensões utilizadas para a representação da variabilidade nos modelos gerados. A seção 3 apresenta a validação da arquitetura proposta. A seção 4 apresenta trabalhos relacionados. Por fim, a seção 5 apresenta as conclusões.

2. O Processo de Desenvolvimento da Arquitetura

O conceito de arquitetura de linha de produtos é recente e ainda existe uma carência de técnicas que venham a facilitar o processo de desenvolvimento da arquitetura e de seus componentes. Algumas abordagens existentes são a Systhesis que foi documentada pelo *Software Productivity Consortium* [3], *Family-Oriented Abstraction, Specification and Translation* (FAST) [4], *Product Line Software Engineering* (PuLSE) do Centro de Fraunhofer [5] e o *Feature-Oriented Domain Analysis* (FODA) documentado pelo SEI [6].

O processo proposto neste trabalho envolve a exploração do domínio por meio do modelo de referência e da arquitetura genérica para WfMS da WfMC, o método Catalysis [7] e a linguagem Rapide [8] [9] para simulação e avaliação da arquitetura. O método Catalysis foi utilizado porque é uma abordagem geral de desenvolvimento baseado em componentes que envolve conceitos de arquitetura de software, *frameworks* e *patterns*. Catalysis é baseado em UML [10]. Assim, o objetivo é aplicar um método de propósito geral, examinando as alterações necessárias ao invés de se criar um método específico para linha de produtos. As abordagens de engenharia de domínio fornecerem recursos para representação de variabilidade, porém são mais deficientes na representação e implementação de arquiteturas de componentes. Assim, os métodos de DBC surgem como uma boa solução para as fases de projeto e implementação da linha de produto. Iniciativas similares já podem ser encontradas na literatura, tais como Kobra [11] e GenVoca[12].

O processo proposto pelo Catalysis que foi seguido neste trabalho para o desenvolvimento de uma arquitetura de linha de produtos é composto das seguintes fases: análise dos requisitos, especificação do sistema, projeto da arquitetura e projeto interno dos componentes. As seções seguintes apresentam uma caracterização dessas fases, bem como apresenta os artefatos gerados em cada uma delas.

2.1 Especificação de Requisitos

Para a elaboração de uma arquitetura de linha de produtos para WfMS deve-se primeiramente elaborar o modelo do domínio representando objetos e ações do domínio em questão. Nesta fase é possível identificar aspectos comuns entre os produtos e os pontos de variação.

O modelo de referência e a arquitetura genérica para WfMS da WfMC serviram de base para extrair o conjunto de funcionalidades necessárias para os produtos de *workflow*. O modelo e a arquitetura indicaram quais os potenciais componentes de um WfMS e suas interfaces. Eles foram projetados para que fabricantes de produtos e componentes pudessem desenvolver diversas partes de WfMS independentemente e integrá-las conforme a demanda das organizações. Além disso, são definidas características importantes para que produtos específicos possam ser construídos de acordo com as especificidades dos negócios modelados, por exemplo, *workflow* para produção de software ou *workflow* para administração financeira. A Figura 1 apresenta a arquitetura genérica para WfMS proposta pela WfMC. A partir destes modelos, foram analisadas as possibilidades de extensão da arquitetura e análise de semelhanças e variabilidade. O padrão Process Manager [13] também foi utilizado no desenvolvimento da arquitetura proposta. Este padrão foi desenvolvido para a definição de um gerenciador de processos em PSEEs (Ambiente de Engenharia de Software Orientado a Processo).

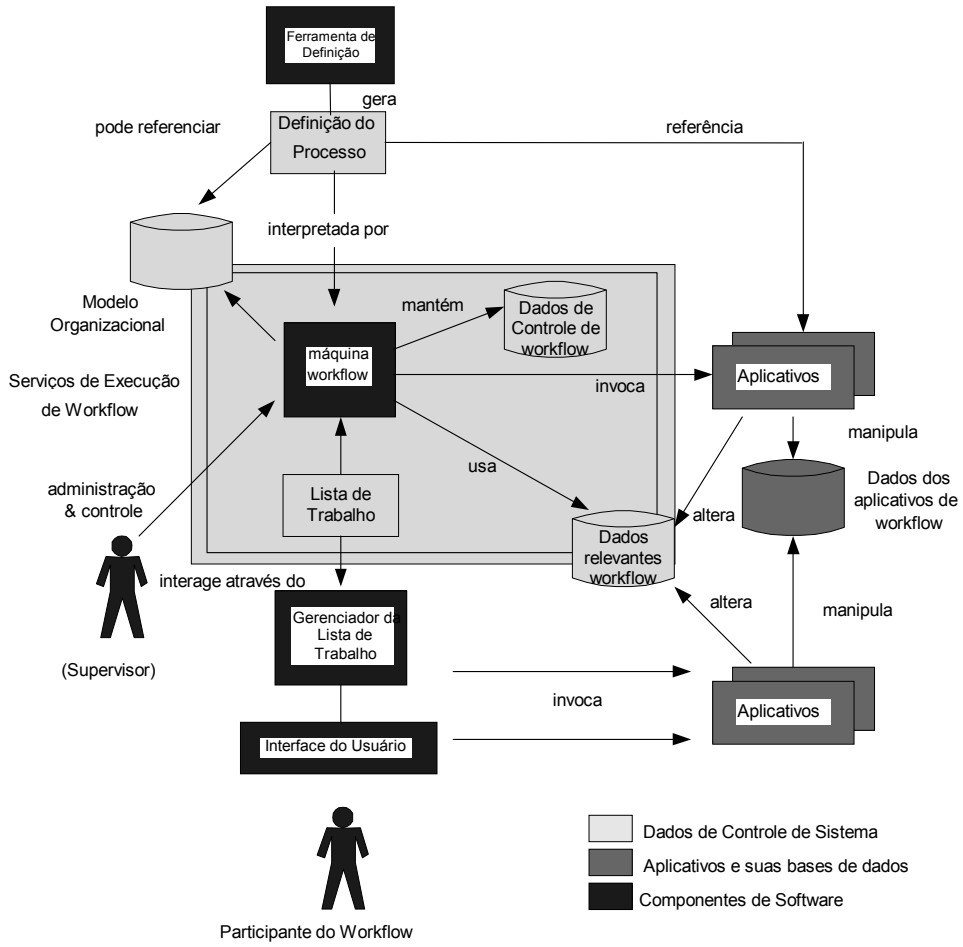

Figura 1 – Arquitetura Genérica para Sistemas de Gerenciamento de *Workflow* [2].

A representação do modelo do domínio pode ser feita utilizando-se objetos e ações, em um nível de abstração independente da eventual solução de software que venha a ser encontrada para o problema. Diagramas de casos de uso da UML dão suporte a esta etapa. As Figuras 2, 3 e 4 apresentam os diagramas de caso de uso para um WfMS. Os objetos representados nos diagramas são o Gerente de Arquitetura de *Workflow*, o Gerente de *Workflow* e o Usuário do *Workflow*, respectivamente.

Durante o desenvolvimento de uma linha de produtos, um dos pontos principais é abstrair e representar as variabilidades associadas à arquitetura e aos seus componentes. A primeira representação de variabilidade ocorreu nesta etapa, nos diagramas de casos de uso. Nesta fase, a notação seguida foi a de representação de variabilidade em casos de uso proposta por Jacobson et al. [14] que, sugeriu a utilização do esteriótipo <<extend>> para representar aspectos de variação em casos de uso. O caso de uso estendido recebe uma marca, representada por um círculo preenchido para indicar variabilidade, como pode ser observado nas Figuras 2, 3 e 4. Como exemplo, pode-se observar na Figura 2 o caso de uso Definir Arquitetura possui um ponto de variação, que representa neste caso a característica opcional de permitir alteração dinâmica da arquitetura. O mesmo pode ser observado para o caso de uso Definir tipo Ferramenta, que possui duas extensões: tipo interna e tipo externa. Outras representações de variabilidade foram necessárias nos modelos definidos na fase de especificação do sistema, como serão apresentadas mais adiante na seção 2.2.

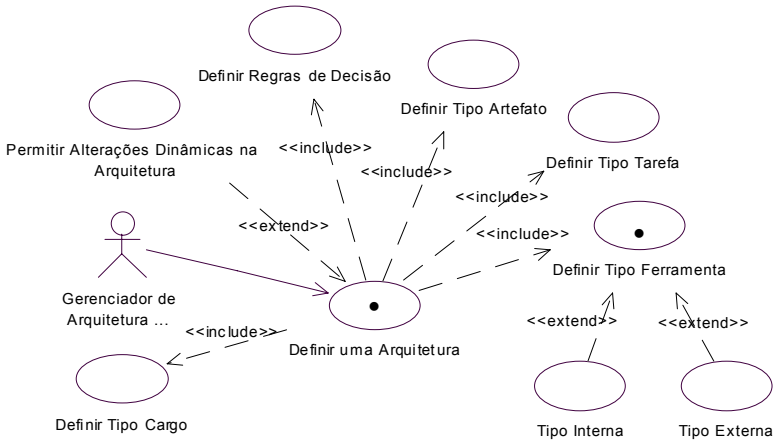

Figura 2 – Diagrama de Caso de Uso do Gerente de Arquitetura de *Workflow*.

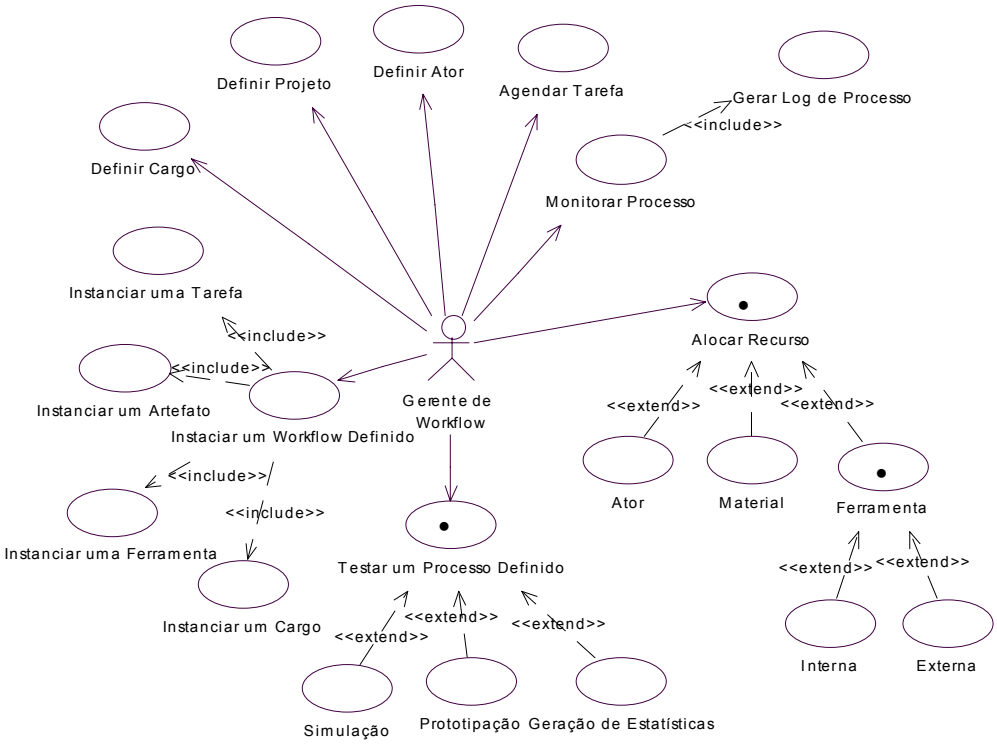

Figura 3 – Diagrama de Caso de Uso do Gerente de *Workflow*.

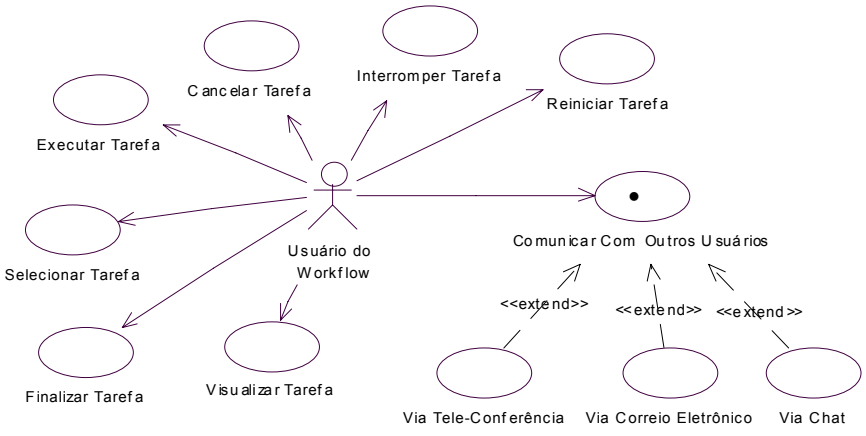

Figura 4 – Diagrama de Caso de Uso do Usuário do *Workflow*.

2.2. Especificação do Sistema

Nesse estágio do desenvolvimento deve-se tratar da 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. Tipos são especificações de comportamento dos objetos que documentam e mostram somente a visão externa de um determinado objeto. O artefato mais importante extraído neste estágio é o diagrama estático de tipos.

Outras representações de variabilidade foram necessárias nos modelos definidos nesta fase do processo, como por exemplo, para o diagrama estático de tipos apresentado na Figura 5. Morisio [15] estendeu a notação UML com um esteriótipo de variabilidade, indicado pela notação <<V>>. Como o esteriótipo <<V>> foi usado no contexto da notação orientada a objetos, este esteriótipo está relacionado aos conceitos como especialização e agregação. Como se pode verificar na Figura 5, o <<V>> indica tipos que podem variar conforme as características dos produtos. Por exemplo, o esteriótipo <<V>> foi usado para representar variabilidade no tipo recurso, que pode ser especializado em *material, ator* e *ferramenta*. O tipo *ferramenta* por sua vez, pode ser especializado em *interna* e *externa*.

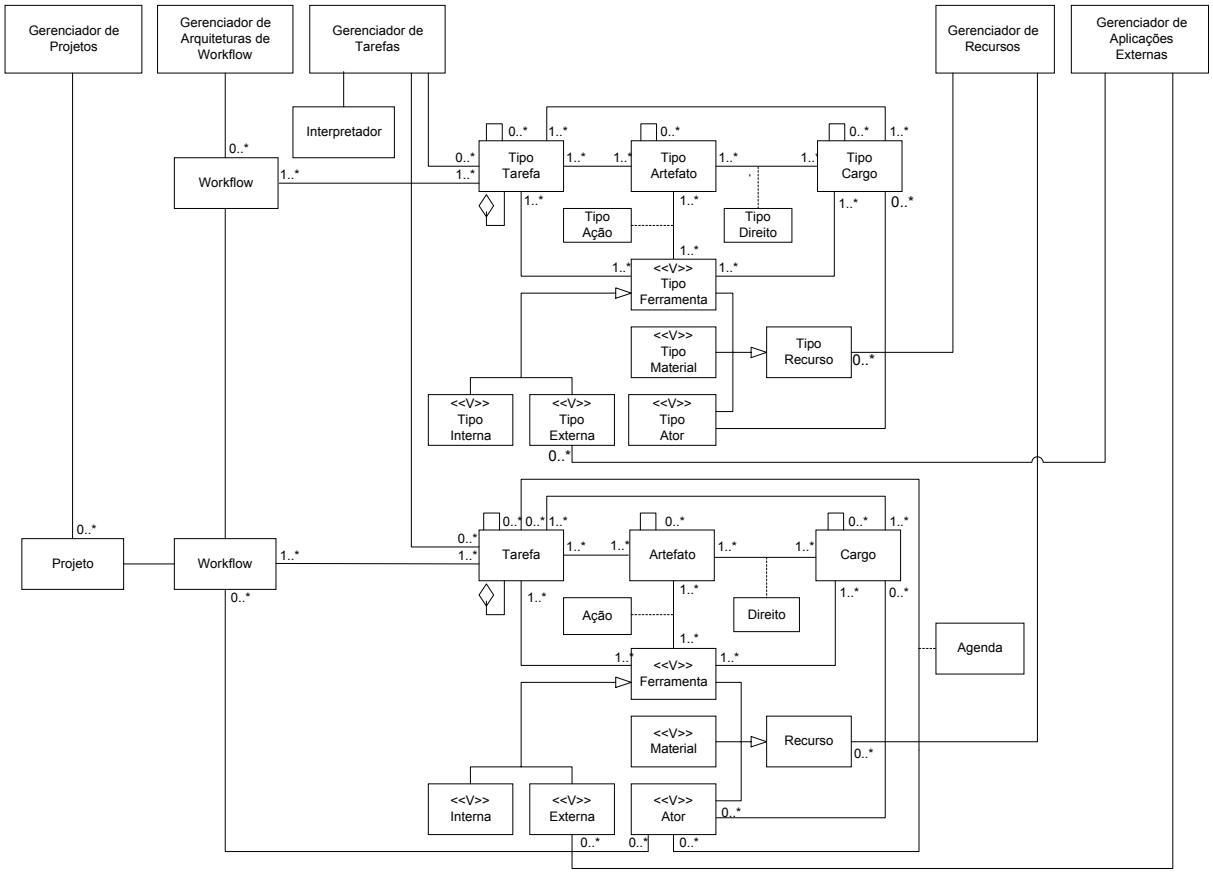

Figura 5 – Diagrama Estático de Tipos para Sistemas de Gerenciamento de *Workflow*.

2.3 Projeto da Arquitetura

Após a modelagem de tipos, uma seqüência de refinamentos desde os níveis mais altos até o nível de componentes deve ser realizado. Catalysis identifica os pacotes como a unidade de decomposição de mais alto nível, considerando que eles representam uma parte do sistema que pode ser tratada de forma independente com o explícito estabelecimento de dependências do restante do sistema. Esses pacotes podem ser obtidos através da análise e refinamento do modelo de negócios e

seus relacionamentos podem ser feitos através de um esquema de importação. O particionamento dos pacotes segue a abordagem de camadas verticais e horizontais proposta por Catalysis. As camadas verticais visam refletir o fato de que usuários diferentes têm diferentes visões de um sistema ou negócio. Os usuários aqui são o projetista de *workflow*, que define a arquitetura do *workflow*; o gerente de *workflow*, responsável pelas atividades de supervisão e administração e, por fim, o usuário que executa as tarefas do *workflow*. Essas três visões são representadas pelos pacotes *Gerenciador de Arquitetura de Workflow*, Gerenciador de *Workflow* e *Gerenciador de Execução do Workflow*, respectivamente. A Figura 6 apresenta o diagrama de camadas verticais de alto nível.

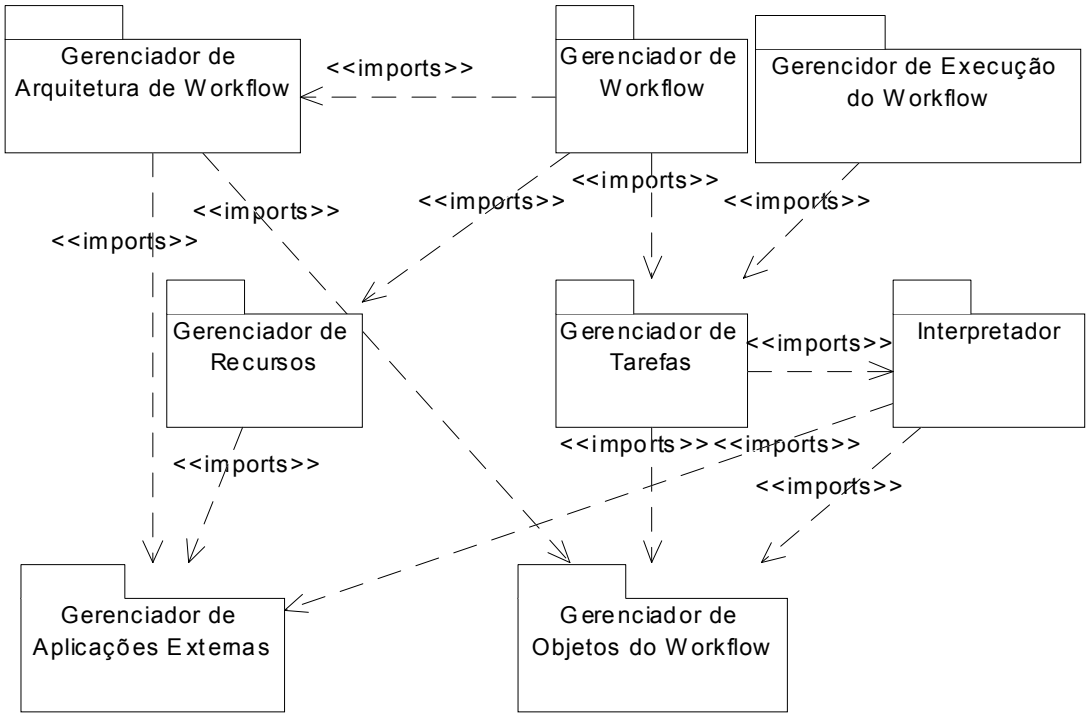

Figura 6 – Diagrama de Camadas Verticais de Alto Nível.

As camadas horizontais visam organizar os pacotes em vários níveis, desde as ações de mais alto nível até as de infra-estrutura. A divisão deve buscar a otimização da importação de pacotes. O relacionamento entre os elementos do diagrama de pacotes e a arquitetura genérica dos WfMSs são representados pelos pacotes Gerenciador de Arquitetura de *Workflow*, Gerenciador de *Workflow*, Gerenciador de Execução do *Workflow*, Gerenciador de Recursos, Gerenciador de Tarefas, Interpretador, Gerenciador de Aplicações Externas e Gerenciador de Objetos do *Workflow*.

A partir do diagrama de camadas verticais de alto nível, foi possível projetar o Diagrama de Camadas Verticais. A construção deste modelo finaliza o processo de particionamento em componentes, apresentando os pacotes definidos anteriormente pelo Diagrama de Camadas Verticais de Alto Nível juntamente com a especificação dos tipos de cada pacote e o relacionamento entre eles. A Figura 7 apresenta o pacote *Gerenciador de Tarefas*, que é parte do Diagrama de Camadas Verticais, juntamente com a especificação de tipos deste pacote. O diagrama completo não é mostrado por questão de espaço.

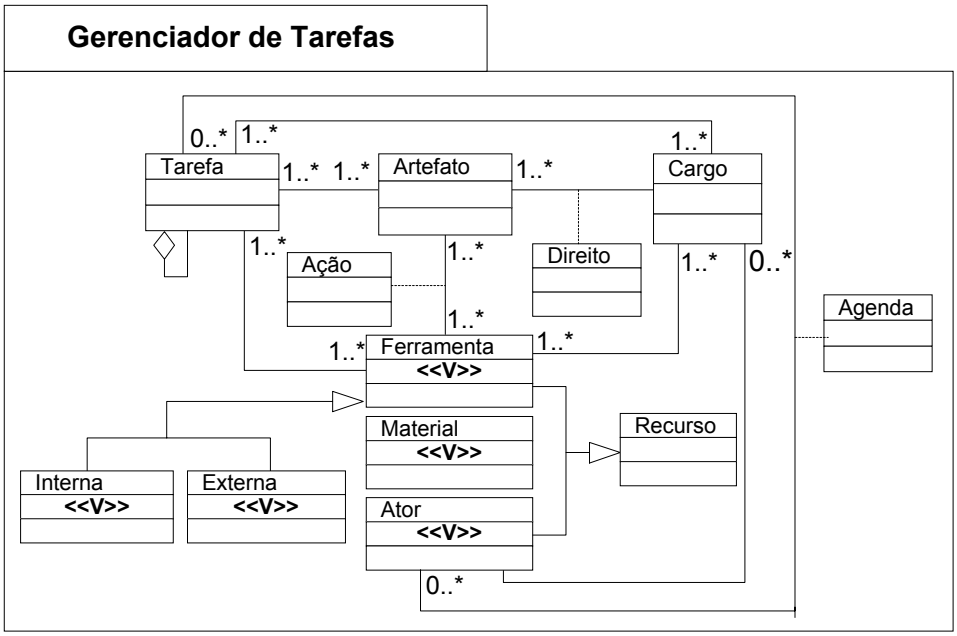

Figura 7 – Especificação do Pacote Gerenciador de Tarefas

2.3.1 Arquitetura de Componentes

O Diagrama de Camadas Verticais apresenta-se como o resultado da identificação e especificação de componentes. Os componentes são representados através de pacotes genéricos, constituídos por tipos e relacionamentos. A arquitetura de componentes para WfMS foi então projetada a partir deste diagrama, conforme apresentado na Figura 8.

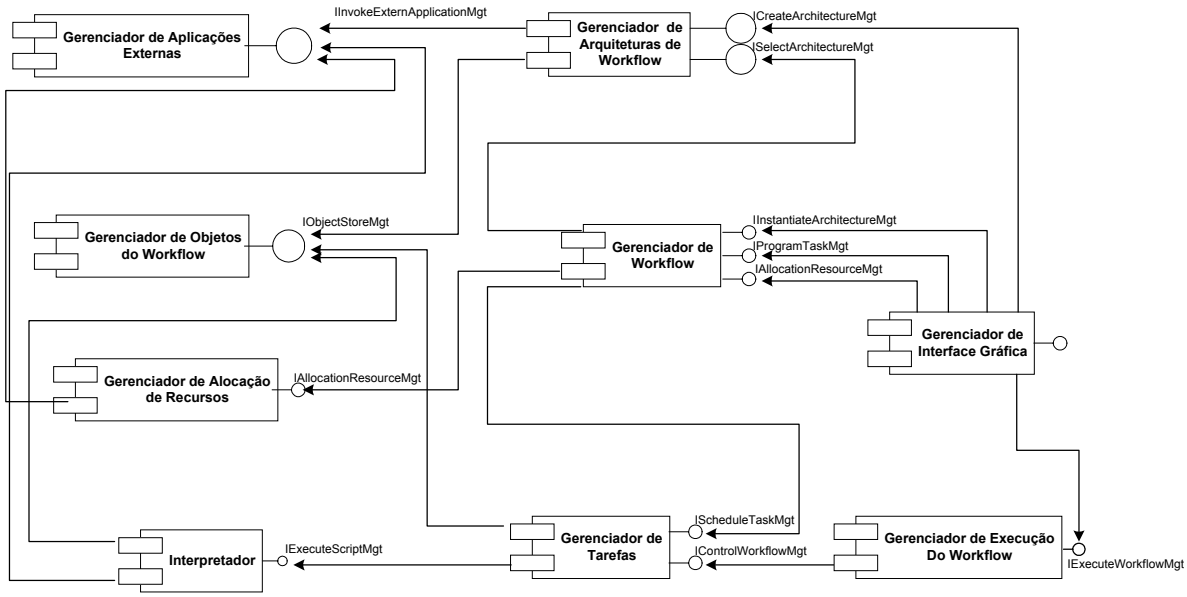

Figura 8 – Arquitetura de Componentes para WfMS.

Uma descrição mais detalhada sobre a funcionalidade dos componentes da arquitetura e aspectos de variabilidade associados a estes componentes serão apresentados mais adiante na seção 2.3.2. Os itens seguintes trazem uma descrição sucinta de cada um dos componentes da arquitetura proposta com suas respectivas interfaces e métodos:

Gerenciador de Interface Gráfica: Este componente é responsável pela interação com o usuário do WfMS.

Gerenciador de Arquitetura de Workflow: Este componente é responsável pelo controle e gerenciamento da construção e manutenção de arquiteturas de *workflow*.

ICriarArquiteturaMgt

SolicitarConexão()	Permite a conexão de um usuário
CriarObjeto()	Cria um novo objeto de definição de *workflow* (ex. um ator, cargo, uma ferramenta)
ExcluirObjeto()	Exclui um objeto de definição de *workflow*
RecuperarObjeto()	Recupera um objeto de definição de *workflow*
InserirAtributoObjeto()	Insere um atributo específico em um objeto de definição
DeletarAtributoObjeto()	Exclui um atributo específico em um objeto de definição
ArmazenarArquitetura()	Salva uma arquitetura de referência

IselecionarArquiteturaMgt

SelecionarArquitetura()	Seleciona uma arquitetura de *workflow*

Gerenciador de Workflow: Este componente é responsável pela criação e gerenciamento de *workflows*.

IInstanciarArquiteturaMgt

SolicitarConexão()	Permite a conexão de um usuário
SelecionarArquitetura()	Seleciona uma arquitetura de *workflow* para instanciação
InstanciarObjeto()	Instancia objetos definidos na arquitetura de *workflow* selecionada

IProgramarTarefaMgt

SelecionarProjeto()	Seleciona projeto a ser executado
SelecionarTarefa()	Seleciona tarefa a ser codificada
SelecionarCargo()	Seleciona cargo ao qual a tarefa a ser codificada está associada
SelecionarAtor()	Seleciona ator para a execução da tarefa
ProgramarTarefa()	Programa uma determinada tarefa

ISelecionarRecursoMgt

SelecionarRecurso()	Seleciona recursos que serão usados na execução de uma determinada tarefa

Gerenciador de Execução do Workflow: este componente apoia a interação com os usuários do WfMS. É através dele que os usuários identificam as suas tarefas no *workflow*.

IExecutarTarefaMgt

SolicitarConexão()	Permite a conexão de um usuário
SelecionarTarefa()	Seleciona uma determinada tarefa
ExecutarTarefa()	Executa uma determinada tarefa
VisualizarTarefa()	Mostra as tarefas pertencentes a um determinado usuário de acordo com o seu cargo

Gerenciador de Tarefas: Este componente é responsável pelo controle e gerenciamento das tarefas e ações a serem realizadas no *workflow*.

IAgendarTarefaMgt

SolicitarConexão()	Permite a conexão de um usuário
AgendarTarefa()	Agenda uma determinada tarefa

IExecutarTarefaMgt

SelecionarTarefa()	Seleciona uma determinada tarefa
ExecutarTarefa()	Executa uma determinada tarefa
CancelarTarefa()	Cancela uma determinada tarefa a ser executada ou que já está em execução por um determinado usuário
InterromperTarefa()	Interrompe uma determinada tarefa
ReiniciarTarefa()	Reinicia uma determinada tarefa
FinalizarTarefa()	Finaliza uma determinada tarefa
VisualizarTarefa()	Mostra as tarefas pertencentes a um determinado usuário de acordo com o seu cargo

Gerenciador de Alocação de Recursos: Este componente é responsável pela alocação de recursos necessários (ex. alocação de atores, alocação de ferramentas, alocação de artefatos e a definição das datas de execução das tarefas) na realização dos projetos.

IAlocarRecursoMgt

SolicitarRecurso()	Solicita um determinado recurso (ex. ferramenta, ator)
LiberarRecurso()	Libera um determinado recurso para efetiva utilização
ExcluirRecurso()	Exclui um determinado recurso
RecuperarEstadoRecurso()	Recupera o estado de um determinado recurso (ex. reservado)

Gerenciador de Aplicações Externas: Este componente é responsável pelo gerenciamento das aplicações externas invocadas durante a definição e execução de um *workflow*.

IinvocarAplicaçõesExternasMgt

SelecionarAplicaçãoExterna()	Seleciona uma aplicação externa
InvocarAplicaçãoExterna()	Invoca uma aplicação externa

Gerenciador de Objetos do Workflow: Este componente é responsável pelo relacionamento com o sistema gerenciador de banco de dados.

IArmazenarObjetoMgt

InserirObjeto()	Insere um novo objeto de *workflow*
AtualizarObjeto()	Atualiza um objeto de *workflow*
ExcluirObjeto()	Exclui um objeto de *workflow*
SelecionarObjeto()	Seleciona um objeto de *workflow*

Interpretador: Interpretas as tarefas definidas para que possam ser executadas.

IExecutarScriptMgt

ExecutarScript()	Executa *Script*
InvocarAplicaçãoExterna()	Invoca uma aplicação externa

2.3.2 Especificação da Arquitetura com CORBA

Segundo o processo seguido pelo Catalysis, após a modelagem da chamada arquitetura lógica (Figura 8), deve-se tomar decisões sobre os mecanismos de implementação a serem utilizados, como por exemplo, o *middleware*. Estas decisões levam ao projeto da arquitetura técnica, conforme apresentada na Figura 9. Para a arquitetura de linha de produtos proposta foi considerado o ORB (*Object Request Broker*) do CORBA [16] como *middleware*, que ocupa o centro da figura. Os componentes são estruturados em camadas de modo a minimizar as importações.

Os componentes da arquitetura possuem interfaces padronizadas e bem definidas através da linguagem IDL (*Interface Definition Language*) do CORBA. Isso faz com que os componentes

possam ser acessados dentro da arquitetura independentemente da linguagem de programação, sistema operacional e rede utilizados.

As comunicações entre os componentes são feitas através do ORB e o componente requisitado atende ou não à requisição, baseando-se em restrições de comunicação que são definidas para cada componente da arquitetura. Por exemplo, o componente *Gerenciador de Interface Gráfica* comunica-se apenas com os componentes *Gerenciador de Arquitetura de Workflow*, *Gerenciador de Workflow* e *Gerenciador de Execução do Workflow*. Qualquer solicitação de comunicação de outro componente da arquitetura com o componente *Gerenciador de Interface Gráfica* será recusada porque existem restrições de comunicação definidas no componente requisitado.

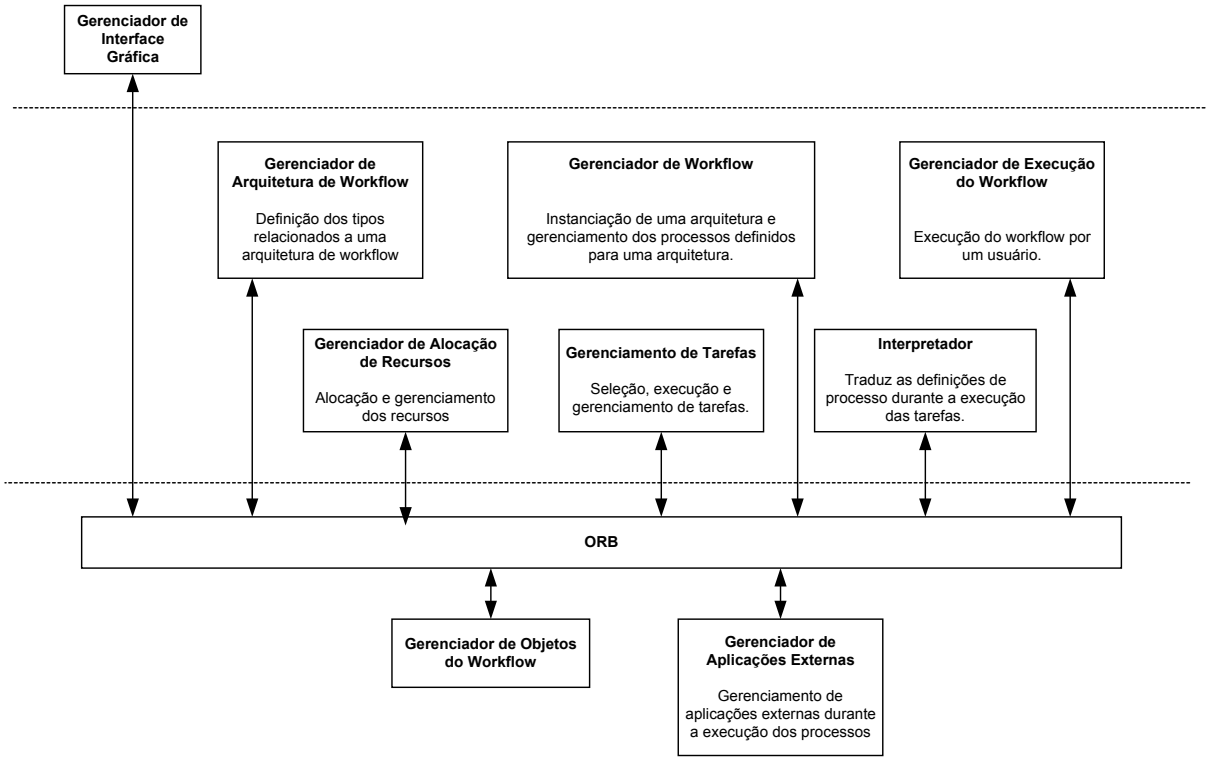

Figura 9 – Arquitetura de Linha de Produtos para WfMS proposta.

Uma descrição mais detalhada de cada um dos componentes da arquitetura, incluindo aspectos de variabilidade, é apresentada a seguir:

Gerenciador de Interface Gráfica: este componente é responsável pelo gerenciamento da interface com o usuário do sistema. O ponto de variação deste componente está no fato de que a interface com o usuário pode ocorrer via interface gráfica convencional ou via *browser*.

Gerenciador de Arquitetura de Workflow: este componente é responsável pelo controle e gerenciamento da construção e manutenção de arquiteturas de *workflow*. A implementação deste componente suporta as funções relacionadas à definição das arquiteturas de *workflow* e os tipos de objetos relacionados a esta. A utilização deste componente torna a definição de *workflows* mais flexível, visto que desta forma, os tipos de objetos não são pré-fixados. Os pontos de variação associados a este componente indicam o tipo de recurso que pode ser utilizado. Este tipo recurso pode ser especializado em ator, ferramenta e material e o tipo de ferramenta pode ser especializado em interna e externa. Diferentes linguagens de programação de processos podem ser apoiadas para a definição das tarefas, que posteriormente serão executadas pelo interpretador.

Gerenciador de Workflow: este 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. Não existem pontos de variação associados a este componente.

Gerenciador de Execução do Workflow: este 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*. Não existem pontos de variação associados a este componente.

Gerenciador de Tarefas: este componente é responsável pelo controle e gerenciamento das tarefas e ações a serem realizadas. É por meio deste os usuários interagem com o sistema gerenciador de *workflow*. O Gerenciador de Tarefas permite que os usuários identifiquem suas tarefas geradas pelo Gerenciador de *Workflow*. Os pontos de variação deste componente são: os tipos de recursos que podem ser utilizados podem ser material, ferramenta e ator; a ferramenta utilizada pode ser do tipo interna ou externa; as tarefas podem ter prioridades de execução diferentes e, os algoritmos utilizados para escalonamento das tarefas também podem variar.

Gerenciador de Alocação de Recursos: este componente é responsável pela alocação de recursos (atores, ferramentas ou material). Além da variabilidade associada ao tipo de recurso e ao tipo de ferramenta, as políticas de alocação de recurso podem variar.

Gerenciador de Aplicações Externas: este componente é responsável pelo gerenciamento das aplicações externas invocadas durante a definição do processo e execução das tarefas. Pontos de variabilidade incluem as diferentes formas de adaptação da aplicação externa ao sistema gerenciador de *workflows*.

Gerenciador de Objetos do Workflow: este 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 pertencentes à arquitetura de componentes proposta 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. Pontos de variação deste componente incluem os adaptadores para os gerenciadores de bancos de dados.

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

Nesta fase do processo, os componentes da arquitetura estão especificados com suas interfaces definidas. Segundo o processo seguido para este trabalho, a etapa seguinte consiste no projeto interno dos componentes da arquitetura. Porém, a implementação desses componentes está fora do contexto deste trabalho. Dentre os trabalhos relacionados a este que está sendo desenvolvido, pode-se destacar a implementação do componente agenda de tarefas por Tanaka [17] e a proposta de um *enterprise framework* para WfMS proposto por Gimenes [18].

3. Validação da Arquitetura Proposta

Bosch [19] identificou quatro formas de validação de uma arquitetura de linha de produtos: uso de cenários, simulação, modelos matemáticos e avaliação baseada em experiências passadas. A seção 2 descreveu o processo de obtenção da arquitetura de linha de produtos para sistemas de gerenciamento de *workflow*. Este processo é baseado no método Catalysis [7], que utiliza a UML como notação. Assim, como na UML [10] não existem recursos para simulação da arquitetura e para que pudéssemos avaliar a arquitetura proposta sem nos aprofundar em detalhes de implementação optamos por utilizar uma ADL (*Architecture Description Language*) para especificar e simular a arquitetura. ADLs [20] são linguagens focadas em representar estruturas de mais alto nível, ao invés de se ater a detalhes de implementação. A ADL Rapide [8] foi identificada como adequada para especificar a arquitetura de linha de produtos para sistemas gerenciadores de *workflow*. Esta linguagem foi escolhida por ser caracterizada como uma linguagem de descrição arquitetural com finalidade geral, que permite modelar interfaces de componentes e seus comportamentos externamente visíveis, e ainda por ser uma linguagem que permite simulação e modelagem do comportamento dinâmico descrito por uma arquitetura.

A simulação foi baseada na seleção de cenários relevantes para os sistemas de gerenciamento de *workflow*. Foram elaborados diagramas de seqüência para representar as interações para cenários específicos de uma certa atividade do sistema, que especificam a comunicação entre os componentes. Esses diagramas serviram de base para realizar a simulação da arquitetura de linha de produto proposta. Dessa maneira, a arquitetura é executada conforme diferentes cenários para simular o comportamento do sistema. Cada cenário representou uma das visões dos diferentes usuários do sistema. Esses usuários são: o gerente de arquitetura de *workflow*, que define a arquitetura de *workflow*; o gerente de *workflow*, responsável pelas atividades de supervisão e administração e, por fim, o usuário que executa as tarefas do *workflow*. Além disso, a modelagem do comportamento dinâmico descrito pela arquitetura de componentes proposta, possibilitou avaliar a comunicação entre os componentes e o efeito geral das funcionalidades do sistema, livrando os detalhes de implementação para a fase de aplicação da arquitetura.

Porém, a simulação não foi suficiente para validar completamente o modelo proposto. A validação completa ainda depende do estabelecimento de critérios apropriados que permitam quantificar o comportamento do sistema real. A chave para a validação baseada em simulação é criar testes que exercitem um erro, de modo a torná-lo aparente para o usuário. Atualmente, testes são criados manualmente e direcionados para casos de teste particulares ou são gerados de modo aleatório na esperança de alcançar situações não testadas de modo probabilístico. Ambos os métodos falham em fornecer uma medida confiável de que um sistema complexo tenha sido testado de modo adequado. Portanto, a simulação, por sua vez, não faz com que a atividade de teste seja facilmente avaliada e quantificada. Dessa maneira, uma outra forma de se validar modelos arquiteturais é através da aplicação do critério Análise de Mutantes, da técnica de teste baseada em erros, que permite avaliar a atividade de teste de forma quantitativa, através do escore de mutação.

4. Trabalhos Relacionados

Não existe um entendimento comum da relação entre reuso, domínios, linhas de produtos, arquitetura de software e *frameworks* [21]. Acredita-se que a controvérsia surge no nível de abstração que cada um desses conceitos representa. O termo linha de produtos parece mais adequado para o contexto industrial. Os conceitos de engenharia de domínio são amplamente utilizados no enfoque de linha de produto.

Os métodos conhecidos de análise de domínio [22], [6], [23], [14] buscam identificar conceitos e funcionalidades requeridas para uma família de produtos e representa-los através de um modelo genérico, que é então refinado pelos estágios subseqüentes de construção de uma infraestrutura de reutilização. Esses métodos utilizam o conceito de *features* para representar as funcionalidades comuns do domínio e também para representar aspectos de variabilidade. Catalysis,

apesar de não ser um método de engenharia de domínio, apresenta o conceito de tipo (análogo ao conceito de *feature*), que seria descrição de alguma funcionalidade da aplicação em termos de seu comportamento externo.

A abordagem de linha de produtos está também diretamente relacionada com as técnicas de desenvolvimento de *frameworks*. Gimenes et al. [17] [18] propõe técnicas de definição de *frameworks* no contexto de WfMS que ofereceram *guidelines* para a construção da arquitetura proposta neste trabalho. Essas técnicas envolvem a utilização do conceito de *framework* de modelos do Catalysis como base para geração de componentes.

5. Conclusões

Este artigo apresenta o processo seguido para o projeto de uma arquitetura de linha de produtos para sistemas de gerenciamento de *workflow*. As extensões utilizadas para representar variabilidade durante o processo de desenvolvimento foram aplicadas com base nos diagramas de caso de uso propostos por Jacobson [14] e extensão da notação UML proposta por Morisio [15].

O método Catalysis [7] foi utilizado para guiar o processo de extração e formalização da arquitetura de componentes dos WfMS, uma vez que os documentos da WfMC [2] especificam a arquitetura de um WfMS por meio de diagramas informais. Porém, para avaliação da arquitetura proposta optou-se por utilizar a ADL [8] Rapide que possui um conjunto de ferramentas de suporte a simulação de arquiteturas. A escolha desta linguagem permitiu a simulação sem que houvesse necessidade de aprofundamento em detalhes de implementação.

A principal contribuição deste trabalho é a arquitetura de linha de produtos para sistemas de gerenciamento de *workflow*. Pode-se destacar também contribuições para a formalização de um processo de desenvolvimento de arquiteturas de linhas de produtos e também um melhor entendimento dos conceitos e abordagens relacionados à prática de linha de produtos.

Para que se possa gerar produtos a partir da arquitetura proposta, deve-se selecionar os componentes da arquitetura que cumprem os requisitos do produto específico, bem como instanciar as variabilidades associadas aos componentes da arquitetura.

5. Referências

[1] Bass, Len et al. "*Software Architecture in Practice"*. Addison Wesley Longman, 1998. 452 p.

[2] Workflow Management Coalition. "*Workflow Reference Model*". Document number TC00-1003, January 19, 1995. 55 p.

[3] Software Productivity Consortium. "*Reuse-Driven Software Processes Guidebook*", SPC-92019-CMC version 02.00.03 November 1993.

[4] Weiss, D. M. and Chi Tau Robert Lai. "*Software Product-Line Engineering: A Family-Based Software Development Approach*". Addison-Wesley, 1999.

[5] Bayer, J., O. Flege, P. Knauber, et al., "*PuLSE: A methodology to develop software product lines*", Symposium on Software Reusability (SSR99), May 1999.

[6] Kang, K., et al. "*Feature-Oriented Domain Analysis (FODA) Feasibility Study (CMU/SEI-90-TR-21, ADA 235785)*". Pittsburgh, PA: SEI CMU, 1990.

[7] D'Souza, D. F. et al. "*Objects, Components and Frameworks with UML – The Catalysis Approach*". Addison Wesley Publishing Company, 1999.

[8] Computer Science Lab. "*DRAFT Guide to Rapide 1.0 – Language Reference Manuals*", Rapide Desigh Team – Program Analysis and Verification Group. Stanford University, 1997.

[9] Luckham, J. J. et al. "*Specification and Analysis of System Architecture Using* Rapide". IEEE Transactions on Software Engineering, Special Issue on Software Architecture,

vol 21, nº4, Abril 1995.pag. 336 a 355.

[10] Rumbaugh, J. "*The Unified Modeling Language Reference Manual*", Addison-Wesley Pub. Company, 1999.

[11] Atkinson, C. et al. "*Component-Based Product Line Development: The Kobra Approach*". In: 1st INTERNATIONAL SOFTWARE PRODUCT LINE CONFERENCE, 2000, Pittsburgh. Proceedings… Pittsburgh: [s.n], 2000.

[12] Batory, D. "*Product Line Architectures*". Erfurt, Germany. Trabalho apresentado em Smaltalk and Java in Industrie and Ausbildung. 1998.

[13] Gimenes, I.M. S. et al., "*Um Padrão para Definição de um Gerenciador de Processos de Software*", In: II Workshop Ibero Americano de Engenharia de Requisitos Y Ambientes de Software, San José, Costa Rica, Ideas'1999 Memorias, 1999, San José: Istituto Tecnológico de Costa Rica, 1999, v.1, pp. 30-46.

[14] Jacobson, I. et al. "*Software Reuse – Architecture Process and Organization for Business Success*". New York: Ed. Addison-Wesley, 1997.

[15] Morisio, M., Travassos, G.H., Stark, M. "*Extending UML to Support Domain* Analysis", IEEE International Conference on Automated Software Engineering – ASE'00. Grenoble, France, 2000.

[16] Object Management Group. "*COM versus CORBA: A Decision Framework*". Available in http://www.quoininc.com/COM_CORBA.html . Last access dez/2000.

[17] Gimenes, I. M. S., Tanaka, S. e Oliveira, J. P. M., "*An Object Oriented Framework for Task Scheduling*", In: *TOOLS Europe 2000*, 2000, Mont St. Michel, France, Tools 33 Technology of Object-oriented Languages and Systems, USA: IEEE Computer Society Press, 2000, v.1, pp. 383-394.

[18] Gimenes, I. M. S. et al. "*Enterprise Frameworks for Workflow Management Systems*". Software Practice & Experience, 2001, a ser publicado.

[19] Bosch, J., "*Design & Use Of Software Architectures. Adopting and Evolving a Product-Line Approach*", Addison-Wesley, 2000.

[20] Medividovic, N. et al. "*A Classification and Comparison Framework for Software Architecture Description Languages*", IEEE Transactions on Software Engineering, v. 26, jan. 2000.pp 70-92.

[21] Poulin, J. "*Software Architectures, Product Lines, and DSSAs: Choosing the Appropriate Level of Abstraction*", WISR8, 1997.

[22] Kang, K., et al. "*FORM: A Feature-Oriented Reuse Method with Domain-Specific Refrence Architecture*", SEI Technical Report, 1998.

[23] Griss, M., et al. "*Integrating Feature Modeling with RSEB*", 5th International Conference on Software Reuse (ICSR-5), ACM/IEEE, Victoria, Canadá, Junho 1998.