Reutilização de Processos de Desenvolvimento de Software Orientado a Objetos Baseada em Padrões

Francisco M. de Vasconcelos Jr.
fmdevjr@cos.ufrj.br

Cláudia Maria Lima Werner
werner@cos.ufrj.br

Programa de Engenharia de Sistemas e Computação
COPPE/UFRJ
Caixa Postal 68511
CEP 21945-970 - Rio de Janeiro - RJ - Brasil

Resumo

No contexto do Memphis - Ambiente de Desenvolvimento de Software Baseado em Reutilização [Rocha96], é apresentada uma abordagem para a reutilização de processos de desenvolvimento de software orientado a objetos baseada em padrões. Para isso, são apresentados, inicialmente, os aspectos a serem levados em conta na elaboração de um meta-modelo para representação de processos de desenvolvimento e descrita a tecnologia de padrões [Gamma95]. Neste contexto, são discutidos os requisitos gerais da ferramenta de reutilização de processos baseada em padrões.

Abstract

In the context of the Memphis - Reuse-Based Software Development Environment [Rocha96], is presented an approach to the pattern-based reuse of object-oriented software development processes. In this way, initialy we present the aspects to be considered in the building of a meta-model for the representation of software development process, and the tecnology of patterns [Gamma95]. In this context, the general requirements of the pattern-based reuse tool are presented.

1. INTRODUÇÃO

Historicamente, o desenvolvimento de software tem se caracterizado por ser centrado no produto. No entanto, recentemente, pesquisadores e desenvolvedores têm mudado o foco de seus esforços para a dimensão do processo [Yu94], buscando obter uma representação deste que venha a sanar alguns dos problemas comuns em organizações de software de grande porte, tais como [Curtis92]:

- Descrição do processo encontra-se distribuída em quantidade muito grande de papéis;
- Processo descrito não corresponde ao real;
- Processo é descrito em um nível alto demais, não sendo útil na prática;
- Descrição imprecisa, ambígua ou incompreensível; e
- Descrição desatualizada.

A compreensão do processo de desenvolvimento de software em larga escala é extremamente complexa, considerando-se o grande número de tarefas e de recursos humanos e de software envolvidos. Gerenciar estes recursos de forma eficiente torna-se uma tarefa quase impossível de se realizar se o processo de desenvolvimento de software não estiver automatizado, ou mesmo parcialmente automatizado. Além disso, Finkelstein [Finkelstein89] e Nuseibeh [Nuseibeh93] argumentam que a modelagem do processo de desenvolvimento permite um melhor entendimento sobre a maneira como sistemas complexos de software são projetados, construídos, mantidos e estendidos. Finkelstein afirma, categoricamente, que o desenvolvimento de software quando não sofre meramente por uma má coordenação, sofre por uma má compreensão [Finkelstein89].

Atualmente, ambientes de desenvolvimento de software (ADS) buscam abranger todo o processo de desenvolvimento. Tais ambientes oferecem suporte a todas as fases do ciclo de vida, possibilitando a manipulação da representação do software em seus diferentes níveis de abstração e facilitando a gerência do processo de desenvolvimento, através da integração das atividades de gerência às de desenvolvimento. A modelagem do processo incorpora à descrição do software o processo usado para desenvolvê-lo, ou melhor, cria uma descrição mais completa do software, uma vez que, como afirma Osterweil [Osterweil87], "o processo também é software". Curtis apresenta uma análise dos objetivos e vantagens da modelagem de processos [Curtis92]:

- **Facilitar o entendimento e a comunicação:** Requer um modelo do processo, com suficiente informação para sua realização e construído através de uma formalização do conhecimento da organização;
- **Suportar o aperfeiçoamento do processo:** Requer a reutilização de processos efetivos e bem definidos, a comparação de processos alternativos e o suporte à evolução do processo;
- **Suportar a gerência do processo:** Requer um processo definido com suas particularidades e a monitoração, o gerenciamento e a coordenação do processo;
- **Automatizar a orientação do usuário:** Requer um ADS efetivo, que possa prover ao usuário assistência, sugestões e informações; e
- **Automatizar o suporte à execução:** Requer porções automatizadas do processo, o suporte ao trabalho cooperativo, a coleta de métricas e a imposição de regras que assegurem a integridade do processo.

Por envolver todas estas questões, a modelagem de processos é complexa. Como afirma Silva, "a construção de um modelo de um processo pode ser tão complexa quanto a construção do próprio produto de software" [Silva92]. Por essa razão, a possibilidade de reutilização de modelos de processos apresenta uma facilidade importante na complexa tarefa

de elaboração de um processo de desenvolvimento, pois pode considerar processos já bem estabelecidos em determinadas áreas (ou domínios de aplicação), aproveitando todo o conhecimento e a experiência adquiridos. Além disso, a possibilidade de reutilização de processos torna-se um incentivo à busca do aperfeiçoamento dos processos de desenvolvimento da organização.

No entanto, o apoio automatizado de um ADS que permita a modelagem do processo, com a possibilidade de reutilização, não é suficiente para que o estabelecimento de um processo de desenvolvimento seja uma tarefa simples. O estabelecimento de um processo continua sendo a organização de uma grande quantidade de informações, que precisa levar em conta o domínio do problema e a experiência da organização. Mesmo com todo este apoio, ainda é necessário o conhecimento de um especialista para a definição do processo. Além disso, com vistas à obtenção de produtos de qualidade, deve ser seguida uma norma de qualidade para o estabelecimento de processos de desenvolvimento (por exemplo, a norma ISO 9000-3 [ABNT93]). Esta norma fornece diretivas visando o estabelecimento de um processo que possua a capacidade de gerar produtos de qualidade.

Neste sentido, novas técnicas de representação são elaboradas na busca de um melhor aproveitamento do conhecimento dos especialistas. Por exemplo, os padrões [Gamma95] foram elaborados com este objetivo. Os padrões encapsulam problemas recorrentes no desenvolvimento de software, incluindo informações as mais completas possíveis sobre estes problemas. Portanto, além do objeto de reutilização, ou seja, a solução de um dado problema, os padrões trazem informações úteis como, por exemplo, a descrição do problema endereçado, sua motivação e o contexto de aplicação.

O projeto de pesquisa em desenvolvimento na COPPE/UFRJ, financiado pelo CNPq, para a construção de um ADS com ênfase na reutilização, supõe a possibilidade de modelagem de processos e sua reutilização. Neste ambiente, a reutilização considera todos os produtos gerados durante o processo de desenvolvimento do software, bem como, o processo propriamente dito, incluindo a definição de recursos (humanos, software e hardware), atividades, produtos de software e métodos utilizados. Neste ambiente, explora-se a tecnologia de padrões para a representação de conhecimento sobre processos, servindo como base de uma ferramenta de reutilização.

Na seção 2, a seguir, é apresentada uma discussão sobre os conceitos básicos envolvidos na modelagem de processos de desenvolvimento de software e uma descrição resumida sobre a norma ISO 9000-3. Na seção 3, discute-se os padrões. Na seção 4, os requisitos da ferramenta para a reutilização de processos baseada em padrões são apresentados. Por fim, na seção 5 concluímos o trabalho.

2. MODELAGEM DE PROCESSOS DE DESENVOLVIMENTO DE SOFTWARE

A reutilização sistemática é uma meta a ser atingida. Esta reutilização sistemática significa uma abordagem institucionalizada da organização para o desenvolvimento de software, onde componentes são intencionalmente criados ou adquiridos para serem reutilizados [Griss95]. Para que se atinja este nível, torna-se necessário empreender um processo evolutivo, com o esforço da organização no sentido de mudanças no negócio, no processo de desenvolvimento, na gerência e na estrutura organizacional.

Nesta direção, propostas como a do projeto REBOOT [REBOOT94] apresentam uma sistemática em que etapas sucessivas vão sendo buscadas nas áreas de gerência da organização, processo de desenvolvimento, gerência do produto, e marketing e negócio.

Com relação à evolução do processo de desenvolvimento, o modelo CMM [Jones95], da SEI, apresenta uma classificação da maturidade do processo de desenvolvimento de uma organização que pode auxiliar na definição de um caminho a ser seguido. Este modelo classifica o processo de desenvolvimento em cinco níveis, desde um desenvolvimento caótico até um desenvolvimento perfeito, otimizado.

A reutilização de processos de desenvolvimento traz para a organização os benefícios comuns da tecnologia de reutilização, mais especificamente, permite uma maior facilidade na elaboração de novos processos de desenvolvimento, possibilitando o aproveitamento do conhecimento e da experiência adquiridos em processos já bem estabelecidos.

Os objetos de reutilização devem estar representados num repositório de componentes. Para que sejam úteis, é suficiente que os componentes sejam representações de objetos que compõem o processo, ou seja, partes de definições ou de produtos de processos. Alguns exemplos de componentes que podem ser usados diretamente seriam, por exemplo, organização de equipes, definição de atividades e de recursos, bem como produtos gerados durante o desenvolvimento (por exemplo, código, diagramas e documentos). No entanto, o ideal é que estes componentes possuam informações adicionais que os descrevam. Estas informações adicionais facilitam aquele que vem a ser um dos aspectos chaves da reutilização, ou seja, a compreensão dos componentes.

Além disso, para que os produtos de um processo de desenvolvimento de software possuam qualidade, é necessário que a qualidade seja uma preocupação ao longo de todo o processo. Estabelecendo de que forma deve ser realizado o controle da qualidade, surgiram normas de qualidade referentes a processos de desenvolvimento de software, tal como, a norma ISO 9000-3, que indica quando um processo possui a capacidade de gerar produtos de qualidade. Portanto, os componentes do repositório precisam estar de acordo com estas normas de qualidade.

Desta forma, o repositório de componentes deve permitir a representação de componentes que englobem o conhecimento de especialistas num domínio, da forma mais completa possível e de acordo com normas de qualidade de processos estabelecidas.

2.1 Conceitos Envolvidos na Elaboração de um Meta-Modelo de Processo

Para a elaboração de um meta-modelo de processo[1], devem ser considerados os conceitos básicos envolvidos em processos de desenvolvimento de software, que são os seguintes [Dutton93]:

- **Artefatos:** São produtos de software. Podem ser produzidos ou consumidos por uma atividade. São exemplos de artefatos: manuais de qualidade, manuais de revisão, diagramas de fluxo de dados, diagramas de objeto e código fonte;
- **Recursos:** São as pessoas da organização, as ferramentas de software, os equipamentos, ou quaisquer outros recursos necessários à execução da atividade;
- **Papel:** Representa o tipo de trabalho realizado por uma pessoa numa atividade;
- **Atividade:** É uma tarefa ou trabalho a ser realizado, requerendo um ou mais recursos para executá-la. Uma atividade pode consumir ou produzir artefatos. Num processo real, as atividades são divididas em sub-atividades, que por sua vez podem também ser subdivididas. Dependendo da complexidade, uma atividade pode ser subdividida em vários níveis. Outra denominação para esta entidade é *tarefa*;
- **Processo:** Um processo de desenvolvimento é um conjunto organizado de métodos, ferramentas, procedimentos e técnicas, com o fim de desenvolver e manter um software; e

[1] Meta-modelo de processos é um modelo para representação de modelos de processos.

- **Projeto:** Um projeto é a organização de um desenvolvimento de software, que emprega um processo de desenvolvimento e que engloba o conjunto de atividades necessárias, documentos consumidos e produzidos, bem como recursos utilizados.

A norma ISO 9000-3 define os seguintes conceitos relativos a processos [ABNT93]:

- **Software:** Criação intelectual compreendendo os programas, procedimentos, regras e qualquer documentação correlata à operação de um sistema de processamento de dados;
- **Produto de Software:** Conjunto completo de programas de computador, procedimentos e documentação correlata, assim como dados designados para entrega a um usuário;
- **Item de Software:** Qualquer parte identificável de um produto de software, em etapa intermediária ou final do desenvolvimento;
- **Desenvolvimento:** Todas as atividades a serem realizadas para a criação de um produto de software;
- **Fase:** Segmento definido do trabalho[2];
- **Verificação:** Processo de avaliação dos produtos de uma determinada fase, a fim de assegurar a correção e a consistência no que diz respeito aos produtos e normas fornecidos como entrada para a referida fase; e
- **Validação:** Processo de avaliação de software, a fim de assegurar que este atende aos requisitos especificados.

As informações capturadas nos conceitos considerados em um meta-modelo de processo de desenvolvimento de software podem ser organizadas sob diferentes pontos-de-vista. Estes pontos-de-vista, se combinados, fornecem, por hipótese, uma descrição integrada, consistente e completa do processo. Em [Curtis92], é apresentada a seguinte lista contendo os pontos-de-vista mais comuns encontrados na literatura:

- **Funcional:** Representa as atividades a serem realizadas e de que artefatos necessitam;
- **Comportamental:** Representa quando as atividades são realizadas, bem como aspectos de como são realizadas através de, por exemplo, ciclos de retro-alimentação e iterações;
- **Organizacional:** Representa a distribuição na organização da realização das atividades, incluindo recursos responsáveis, meios de comunicação e meios físicos de armazenamento, bem como suas localizações; e
- **Informacional:** Representa que artefatos são produzidos e consumidos pelas atividades, incluindo suas estruturas e relacionamentos.

Os meta-modelos são construídos tendo como base linguagens ou abstrações elaboradas para a representação de informações, que vão determinar algumas de suas características. Por exemplo, na tabela 1 apresenta-se uma lista de abstrações de representação, analisando sua expressividade em relação às informações do processo, organizadas segundo os pontos-de-vista relatados acima. Como se pode notar, nenhuma característica por si só é suficiente para cobrir todas as informações. Inclusive, os meta-modelos encontrados na literatura, normalmente, apresentam linguagens com características classificadas como baseadas em mais de uma destas abstrações. A maioria dos meta-modelos encontrados na literatura adotam a abordagem, considerada necessária, de integrar múltiplos paradigmas representacionais, utilizando mais de um modelo para a captura das informações dos processos, apesar disto trazer uma maior complexidade na elaboração do meta-modelo.

---

[2]Uma fase não implica no uso de qualquer modelo de ciclo de vida específico, nem implica em um período de tempo durante o desenvolvimento de um produto de software.

Base	*Func*	*Comp*	*Organiz*	*Inform*
Linguagens de Programação Procedurais	✓	✓		✓
Análise e Projeto de Sistemas (Incluindo DFD, SADT e Diagramas de Estrutura)	✓		✓	✓
Técnicas de IA (Incluindo Regras e Pré-/Pós-condições)	✓	✓		
Eventos e "Triggers"		✓		
Redes de Petri e Transição de Estado	✓	✓	✓	
Fluxo de Controle		✓		
Linguagens Funcionais	✓			
Linguagens Formais	✓			
Modelagem de Dados (Incluindo E/R, relacionamentos e declarações de dados estruturados)				✓
Modelagem de Objetos (Incluindo classes, instâncias, hierarquias e herança)			✓	✓
Redes de Precedência (Incluindo Redes de Petri e CPM)		✓		

Tabela 1 - Expressividade de Linguagens Usadas Como Base de Meta-Modelos [Curtis92]

Algumas questões referentes à modelagem de processos precisam ser levadas em conta para a elaboração de um meta-modelo. Por exemplo, o *grau de formalidade* que a linguagem de modelagem de processos deve apresentar depende do fim da descrição do processo, e será maior se for necessária sua execução pelo computador, ou menor quando se tratar apenas de informar um agente humano. Uma outra questão diz respeito à *granularidade*, que se refere ao "tamanho" dos elementos dos processos representados no modelo, que deve ser tal que permita a representação de informações tão detalhadas quanto se deseje.

2.2 A Norma ISO 9000-3 [ABNT93]

A certificação de um processo de desenvolvimento através de normas de qualidade fornece evidências que o processo é capaz de gerar produtos de qualidade. A norma ISO 9000-3 fornece diretrizes que se destinam a descrever os controles e métodos sugeridos para a produção de software que atenda aos requisitos do usuário, evitando-se não-conformidades em todos os estágios, desde o desenvolvimento até a manutenção.

A norma ISO 9000-3 prega a implementação de um sistema da qualidade durante o desenvolvimento, ou seja, a implementação de um processo de controle da qualidade dos produtos gerados. O objetivo é que a qualidade seja perseguida ao longo de todo o desenvolvimento, em vez de ser verificada apenas no final, enfatizando, assim, a prevenção mais do que a correção após a ocorrência de problemas.

A norma ISO 9000-3 classifica os documentos em três tipos: 1) descrição do sistema de qualidade a ser aplicado ao ciclo de vida do software; 2) documentos de planejamento, descrevendo o planejamento e a execução de todas as atividades do fornecedor e suas interações com o comprador; e 3) documentos de produto, descrevendo um determinado produto de software.

Estão previstos os seguintes documentos: Documentação de Utilização do Produto de Software, Documentação do Sistema de Qualidade, Especificação de Requisitos do Usuário,

Relatório de Situação da Configuração, Registros, Plano da Qualidade, Plano de Desenvolvimento, Plano de Gestão de Configuração, Plano de Ensaio e Plano de Manutenção.

Métricas são importantes para o gerenciamento do processo, na medida em que permitem o gerenciamento do processo de desenvolvimento. A norma ISO 9000-3 cita dois tipos de métricas, de produtos e de processos, afirmando que não existem indicadores de qualidade aceitos universalmente, mas que deve-se usar algum que possa fornecer bases para comparação e que sejam adequados ao processo usado.

Com relação à medições de produtos, a norma prega que deve-se usar algum tipo de indicador que possa expressar falhas e/ou defeitos. Com relação ao processo, a norma identifica a necessidade de se dispor de indicadores quantitativos da qualidade do processo de desenvolvimento e entrega. Prega-se que as métricas devem refletir o quanto o processo está sendo efetuado em termos do cumprimento da programação para o desenvolvimento e para a qualidade.

A norma ISO 9000-3 prevê diversas atividades para o acompanhamento do desenvolvimento. Algumas estão relacionadas às fases específicas do processo de desenvolvimento, enquanto outras podem ser aplicadas ao longo de todo o processo.

As atividades previstas são de três tipos: 1) atividades de desenvolvimento do software; 2) de planejamento, de controle da qualidade; e 3) de manutenção do software. A atividade de planejamento é realizada em relação ao projeto de desenvolvimento como um todo (*macro-planejamento*), bem como em relação a cada uma das atividades consideradas (*micro-planejamento*).

Não haverá, necessariamente, uma correspondência de um para um entre atividades previstas na norma ISO 9000-3 e atividades realizadas efetivamente durante o desenvolvimento. É de se esperar, entretanto, que em projetos de porte maior as atividades previstas sejam implementadas por diversas atividades efetivas, ocorrendo o oposto em projetos de menor porte.

As atividades previstas são as seguintes: Especificação de Requisitos do Usuário, Planejamento do Desenvolvimento, Planejamento de Ensaios, Planejamento da Manutenção, Planejamento da Qualidade, Projeto, Implementação, Manutenção, Ensaio, Validação, Aceitação, Auditorias Internas da Qualidade, Análises Críticas, Gestão de Configuração, Controle da Qualidade, Medição de Produtos e do Processo, Treinamento.

3. PADRÕES

Os padrões têm sua idéia original atribuída ao arquiteto Christopher Alexander que, no final da década de 60 e na década de 70, propôs uma linguagem de padrões para utilização no projeto arquitetônico, com o objetivo de sanar diversos problemas por ele apontados nestes projetos [Lea94]. A linguagem de padrões (um conjunto de padrões, cada um descrevendo como resolver um tipo específico de problema) proposta por Alexander, inicia num nível de abstração muito grande, explicando como o mundo deveria ser dividido em nações, nações em regiões menores e continua explicando como projetar estradas, estacionamentos, comércio, lugares de trabalho, casas e templos de adoração. Os padrões de Alexander vão apresentando foco com nível de detalhe cada vez maior, passando por uma discussão de como arranjar os quartos e salas de casas até finalmente descrever o tipo de material a ser usado em paredes, a decoração da casa e como tratar a iluminação.

Os padrões de Alexander foram usados como inspiração pela comunidade de desenvolvimento de software orientado a objetos (OO) e, como forma de organização do

conhecimento, estão adquirindo grande importância desde de sua introdução como facilitador da reutilização do conhecimento de especialistas [Gamma95][3].

Os padrões encapsulam soluções de problemas recorrentes no desenvolvimento de software, incluindo informações que determinam e facilitam a aplicação desta solução, tais como contexto, conseqüências de utilização, contra-indicações e sugestões de implementações [Lea94][Gamma95]. A estrutura de um padrão e as informações que inclui têm o objetivo de documentar, de forma mais completa possível, o conhecimento de um especialista.

Um conjunto de padrões com um determinado fim é denominado de uma *linguagem de padrões*, que pode ter uma organização complexa, com relacionamentos de afinidade e até mesmo uma estrutura hierárquica rígida [Lea94].

Na medida em que permitem o registro e organização, de maneira sistemática, do conhecimento acerca de soluções, os padrões figuram-se numa forma de reutilizar a experiência de especialistas, tendo já provado sua utilidade na representação de informações das diversas fases do desenvolvimento de software OO. Desta maneira, são sugeridos, em nosso trabalho para a representação do conhecimento sobre processos de desenvolvimento de software OO.

Na figura 1, encontra-se um gráfico estabelecendo uma escala de evolução de conceitos sobre a representação de dados com que se lida no desenvolvimento de software. O dado seria uma porção do mundo representado. A informação seria porções relacionadas. O conhecimento seria a obtenção de nova porção a partir da informação existente. Finalmente, sabedoria seria a possibilidade de refletir num novo contexto um conhecimento existente. O uso dos padrões permite a representação do conhecimento, de forma direta e sistemática. Até agora, o trabalho de especialistas só estava disponível através de consultas a manuais que explicavam métodos de desenvolvimento, pela investigação de produtos acabados ou consultas aos próprios especialistas. O uso de padrões pretende tornar mais acessível este conhecimento, favorecendo a aplicação da sabedoria.

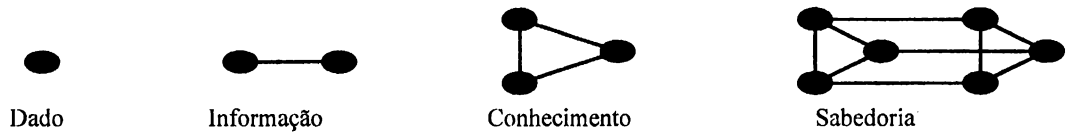

Figura 1 - Evolução de Conceitos do Desenvolvimento de Software [Pressman92]

Neste contexto, o formato[4] dos padrões é importante. Um formato simples para padrões foi apresentado por Lea [Lea95], afirmando ser o formato mais comum dos padrões de Alexander:

SE
você se encontra no CONTEXTO,
por exemplo, EXEMPLOS,
com o PROBLEMA,
vinculado às FORÇAS
ENTÃO
por RAZÕES,
aplicar FORMATO E/OU REGRA DE PROJETO
para construir SOLUÇÃO
levando a NOVO CONTEXTO e OUTROS PADRÕES

[3] Na verdade foi Peter Coad, em [Coad92], quem introduziu os padrões no desenvolvimento OO, mas o trabalho de Gamma et al [Gamma95] é considerado um marco fundamental nesta área.

[4] O formato do padrão pode ser visto como a estrutura de dados elaborada para a representação das informações.

Os padrões de Gamma et al [Gamma95] apresentam um formato genérico, diferindo um pouco do de Alexander. O seu formato possui quatro elementos essenciais:

- **Nome:** Uma identificação, de uma ou duas palavras, que se possa usar para descrever o problema, suas soluções e conseqüências. A nomeação de padrões aumenta o vocabulário de projeto e permite que se projete num nível de abstração mais alto. Além disso, a existência de um vocabulário de padrões facilita sua comunicação e, portanto, sua discussão e evolução;
- **Problema:** Descreve quando o padrão é aplicável, além de explicar o problema e o contexto. Pode descrever problemas específicos de projeto, como por exemplo, de que forma representar algoritmos como objetos. Pode, também, descrever estruturas de objetos e classes que são sintomáticos de um projeto sem flexibilidade. Pode, às vezes, incluir uma lista de condições para a aplicação do padrão;
- **Solução:** Descreve os elementos que compõem o projeto, seus relacionamentos, responsabilidades e colaborações. Não descreve em particular um projeto concreto ou uma implementação, já que um padrão é como um "template", que pode ser aplicado em várias situações diferentes. Ao contrário, um padrão apresenta uma descrição abstrata de um problema de projeto e como um arranjo genérico de elementos (classes e objetos) o soluciona; e
- **Conseqüências:** São os resultados e comprometimentos da aplicação do padrão. Apesar das conseqüências não serem consideradas quando se descreve decisões de projeto, elas são críticas para a avaliação de alternativas, bem como para compreensão de custos e benefícios da aplicação do padrão. As conseqüências em software quase sempre se referem a comprometimentos de espaço e tempo, mas podem abordar também questões de linguagem e implementação. Como a reutilização é sempre um fator importante no projeto orientado a objetos, as conseqüências de um padrão devem incluir o impacto na flexibilidade, extensibilidade e portabilidade do sistema.

Buschmann e Meunier [Buschmann95] observam que uma linguagem de padrões, para ser útil, precisa considerar todas as possíveis restrições relativas à combinação e composição dos padrões que compreende. Apesar de cada padrão endereçar uma questão de projeto em particular, isolada e completa, nenhum padrão pode ser visto como totalmente independente dos outros. Os autores observam que, na realidade, um padrão é parte de uma estrutura maior e que pode conter, estar contido, ou estar interligado com outros padrões, com os quais também interagirá. Portanto, os padrões, quando combinados corretamente, devem complementar-se, de modo a que não sejam introduzidos problemas, mais do que são resolvidos. Ressaltam, ainda, que devem ser elaboradas orientações sob dois aspectos:

- Cada padrão deve especificar com quais padrões pode ser combinado e como, fornecendo ainda instruções sobre sua integração em estruturas maiores, ou sua composição a partir de estruturas menores; e
- Estruturas apresentam complexidade por si só, independentemente das questões de projeto endereçadas pelos padrões que a compõem. Portanto, regras gerais e instruções devem ser especificadas para sua composição e manipulação, independentemente de orientações para seus padrões constituintes.

Os padrões figuram-se numa forma de representação adequada às informações manipuladas ao longo de todo o ciclo de vida. Nas abordagens encontradas na literatura, encontram-se, por exemplo, padrões aplicáveis à gerência e planejamento do processo [Coplien95b], à análise de requisitos [Whitenack95], à análise do sistema [Coad95], ao projeto do sistema [Gamma95][Buschmann95] e à programação [Coplien92].

4. REQUISITOS DE UMA FERRAMENTA PARA A REUTILIZAÇÃO DE PROCESSOS BASEADA EM PADRÕES

O contexto de atuação deste tipo de ferramenta considerado neste trabalho é o Memphis - Ambiente de Desenvolvimento de Software Baseado em Reutilização. Este ADS está em desenvolvimento na COPPE/UFRJ, num projeto integrado financiado pelo CNPq. O Memphis é um dos ambientes instanciados pela Estação TABA [Rocha94].

No contexto do projeto TABA, algumas propostas foram elaboradas considerando a modelagem do processo de desenvolvimento, sendo apresentadas aqui quanto ao aspecto da representação [Aguiar92] [Silva92] [Travassos95].

4.1 Modelagem de Processos

Xamã é como se denomina o sistema especialista de suporte à decisão para planejamento de um ADS da estação TABA [Aguiar92]. Este sistema visa dar suporte ao engenheiro de software na instanciação de um ambiente que seja adequado ao projeto a ser realizado. Em função das respostas fornecidas pelo engenheiro de software a algumas perguntas, Xamã sugere um processo a ser adotado.

Xamã lida com os conceitos de ciclo de vida, método, ferramenta e hardware. O ciclo de vida é visualizado através de um grafo, onde os nós são as atividades e as arestas representam os relacionamentos entre as atividades, fornecendo um sentido de seqüenciamento (uma atividade possui anteriores). Além disso, armazena informações descritivas sobre as atividades (objetivo, data de início, duração), bem como métodos, ferramentas e hardware utilizados em cada atividade. Com relação às ferramentas permite, ainda, que existam ferramentas específicas de métodos.

O trabalho de Silva [Silva92] se baseou na descrição do processo elaborada em [Aguiar92]. Seu elemento principal é um diagrama de estados e transições (pré e pós-condições) baseadas nos atributos dos objetos de software, onde os objetos de software são, por exemplo, os documentos de requisitos, diagramas de projeto e código. Os objetos de software possuem, cada um, um ciclo evolutivo particular, e é a partir da descrição do ciclo de vida de cada objeto que se constrói o modelo do processo.

Em [Travassos95], encontra-se uma proposta de modelagem de processos baseada na de [Silva92]. Em ambas as propostas a preocupação é com a representação do processo, sendo que em [Travassos95] a representação é elaborada usando-se o paradigma orientado a objetos.

A modelagem do processo em [Travassos95] considera os seguintes conceitos: Atividades, Produtos, Pessoas, Recursos e Ferramentas. A modelagem é considerada como sendo constituída de três fases:

- **Trabalho a ser realizado:** Define quais os responsáveis por quais atividades e os recursos e ferramentas necessários;
- **Atividades:** Define as atividades em termos de suas composições (sub-atividades) e dependências; e
- **Produtos:** Define os produtos gerados nas atividades e as ferramentas necessárias.

A abordagem usada para modelar o processo de desenvolvimento no ambiente Memphis basicamente utiliza a descrita em [Travassos95], estando ainda sujeita a refinamento. Este refinamento visa realizar aperfeiçoamentos a partir da análise do meta-modelo segundo os aspectos tratados na seção 2 deste trabalho. Além disso, é necessário estende-lo, provendo informações adicionais, conform tratado na seção 3, tornando a representação do processo de desenvolvimento baseada em padrões.

4.2 Descrição da Ferramenta

A determinação dos componentes de um processo de desenvolvimento não é uma tarefa trivial, uma vez que existem várias possibilidades de combinação, além de ser necessário conhecimento sobre cada componente e como se relacionam entre si [Aguiar92]. No estabelecimento de um processo de desenvolvimento, o engenheiro de software necessita informar, basicamente, quais atividades serão realizadas, quais as pessoas envolvidas e quais os produtos a serem gerados, bem como determinar as responsabilidades das pessoas e em que atividades os produtos serão gerados. Estando estas informações contidas em padrões de processo, a modelagem do processo num ADS se dará através da integração e adaptação das soluções embutidas nestes padrões.

O objetivo da ferramenta aqui descrita é auxiliar o engenheiro de software no estabelecimento de um processo de desenvolvimento, colocando à sua disposição o conhecimento de especialistas acumulado na organização utilizando a tecnologia de padrões.

Uma Linguagem de Padrões para Processos de Desenvolvimento permite que o engenheiro de software tenha em mãos uma gama de soluções para os problemas enfrentados no estabelecimento de um processo de desenvolvimento. Os padrões contêm, por exemplo, informações sobre a organização de equipes, gerência de projeto, produtos típicos e arquiteturas de integração de componentes.

Um exemplo de linguagem de padrões para processos é proposta por Coplien [Coplien95b]. Estes padrões são na maior parte sobre a organização de equipes, tais como, número de pessoas e definição de papéis desempenhados. Os padrões de Coplien apresentam soluções para problemas genéricos de um processo, refletindo o que o autor levantou junto a organizações reconhecidamente eficientes.

A utilização de padrões neste trabalho, visa o suporte ao estabelecimento de um processo em um ADS. Portanto, estes padrões necessitam conter uma quantidade maior de informações sobre processos, que compeendam, por exemplo, informações sobre o domínio aplicável, a definição da rede de atividades a serem realizadas e das atividades em si, os problemas resolvidos, o contexto de aplicação, além da organização de equipes. Além disso, como o ambiente Memphis é representado através de um modelo OO, os padrões que apresentarem como solução um conjunto de classes e objetos relacionados, terão que utilizar as mesmas classes, objetos, e seus relacionamentos, do modelo do Memphis.

No entanto, para tornar efetiva a utilização desses padrões, é preciso que se disponha de uma ferramenta para manipulá-los. Tal ferramenta deve vasculhar o repositório de conhecimento, procurando aqueles padrões que possuem alguma identificação com o problema especificado pelo engenheiro de software.

Na figura 2 encontra-se um esquema do contexto de trabalho da ferramenta. A ferramenta realiza "adaptações" ao processo de desenvolvimento definido inicialmente para o Memphis, ajustando-o de acordo com as necessidades do usuário.

O repositório de conhecimento sobre processos (figura 2), armazena a linguagem de padrões de processo. Ele deve ser populado, em princípio, usando os padrões retirados da definição do processo de desenvolvimento elaborada para o ambiente Memphis [Rocha96], que considera as diretivas estabelecidas pela norma ISO 9000-3 e os padrões de Coplien[Coplien95b]. Uma característica dos componentes deste repositório é que eles têm que ser especialmente construídos para lá figurarem.

A ferramenta apoia o engenheiro de software na adaptação do processo de desenvolvimento à realidade de um projeto específico. No entanto, esta ferramenta também é

útil quando se torna necessária uma adaptação em função da necessidade de solução para algum problema incidente durante o desenvolvimento.

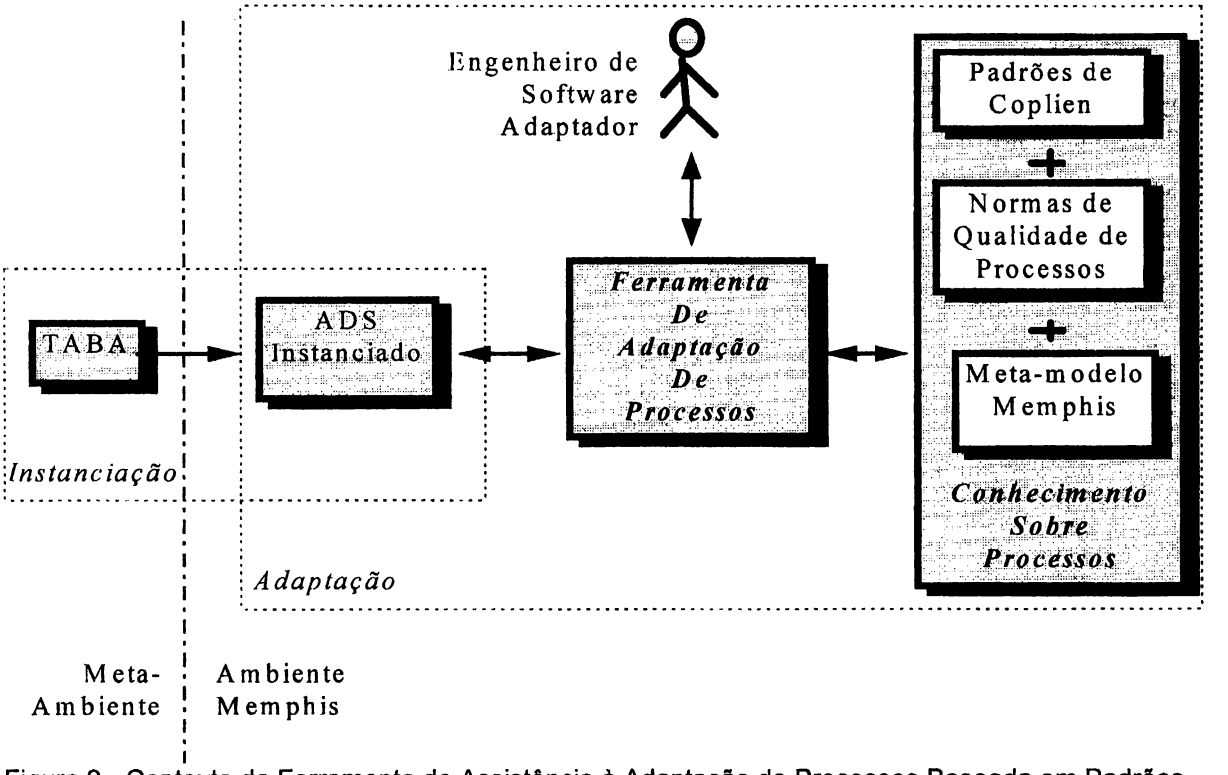

Figura 2 - Contexto da Ferramenta de Assistência à Adaptação de Processos Baseada em Padrões

A partir das diretrizes do engenheiro de software sobre a adaptação que deseja realizar no processo, a ferramenta utiliza as informações contidas nos componentes do repositório (i.e., no catálogo de padrões), fornecendo padrões que apresentem algum conhecimento aplicável.

As atividades de *instanciação* e *adaptação*, representadas na figura 2, podem formar um ciclo. É possível que, eventualmente, o engenheiro de software se decida pelo abandono da atividade de adaptação e realize a instanciação de um novo ambiente, com outras características, buscando obter um ambiente mais próximo do desejado.

O engenheiro de software, usuário típico desta ferramenta, deve possuir conhecimento detalhado sobre o projeto a ser implementado, possivelmente será o próprio gerente do projeto.

5. CONCLUSÃO

ADSs que permitem a modelagem do processo dão suporte à realização organizada e controlada de processos de desenvolvimento de software. A capacidade de reutilizar processos já estabelecidos garante ao engenheiro de software o acesso à experiência acumulada na organização, tornando mais efetivo o suporte dado pelo ambiente. No entanto, este suporte não é suficiente para tornar dispensável o conhecimento do especialista. Buscando tornar mais acessível o conhecimento necessário, apresentamos um ferramenta a ser empregada em tal ambiente para dar suporte à adaptação de um processo de desenvolvimento às especificidades do projeto para o qual será usado.

Os requisitos para esta ferramenta se originam: 1) das informações com que vai lidar, ou seja, processos de desenvolvimento; 2) da técnica de representação destas informações; e 3) do contexto em que vai trabalhar.

Neste sentido, discutimos os aspectos a serem levados em conta para a elaboração de um meta-modelo de processos de desenvolvimento. Apresentamos, então, os padrões, como forma de representação das informações de processos de desenvolvimento. Por fim, identificamos os requisitos de uma ferramenta para adpatação no contexto do ambiente Memphis.

AGRADECIMENTOS

Ao CNPq pelo financiamento a este projeto de pesquisa.

BIBLIOGRAFIA

[ABNT93] Associação Brasileria de NormasTécnicas; *Normas para Qualidade de Processos - Norma ISO 9000-3*; 1993.

[Aguiar92] T.C. de Aguiar; *Um Sistema Especialista de Suporte à Decisão para Planejamento de Ambientes de Desenvolvimento de Software*; Programa de Engenharia de Sistemas e Computação, Tese de Doutorado, COPPE/UFRJ, Março/1992.

[Buschmann95] F. Buschmann e R, Meunier; *A System of Patterns*; em Pattern Languages of Program Design; em [Coplien95a].

[Coad92] P. Coad; *Object-Oriented Patterns*; Communications of the ACM, Setembro/1992.

[Coad95] P. Coad, D. North e M. Mayfield; *Object Models - Strategies, Patterns & Applications*; Prentice-Hall, Yourdon Press Computing Series, 1995.

[Coplien92] J. Coplien; *Advanced C++ Programming Styles and Idioms*; Addison-Wesley, 1992.

[Coplien95a] J. Coplien e D. Schmidt, eds.; *Pattern Languages of Program Design*; Addison-Wesley, 1995.

[Coplien95b] J. Coplien; *A Generative Development-Process Pattern Language*; em [Coplien95a].

[Curtis92] B. Curtis, M.I. Kellner e J. Over; *Process Modeling*; Communications of the ACM, Setembro/1992.

[Dutton93] J.E. Dutton; *Commonsense Approach to Process Modeling*; Software, IEEE, Julho/1993.

[Finkelstein89] A. Finkelstein; *"Not Waving But Drowning": Representation Schemes for Modelling Software Development*; 11th International Conference on Software Engineering, IEEE, 1989.

[Gamma95] E. Gamma, R. Helm, R. Johnson e J. Vlissides; *Design Patterns - Elements of Reusable Object-Oriented Software*; Addison-Wesley, Professional Computing Series, 1995.

[Griss95] M. Griss; *Software Reuse: Objects and Frameworks Are Not Enough*; Object Magazine; Fevereiro/1995.

[Johnson92] R. Johnson; *Documenting Frameworks Using Patterns*; OOPSLA, IEEE, 1992.

[Jones95] D. Jones; *Repeatable Software Development, Part I: Quality Assurance*; Software Development, Maio/1995.

[Lea94] D. Lea; *Christopher Alexander: An Introduction for Object-Oriented Designers*; Software Engineering Notes, ACM SIGSOFT, Janeiro/1994.

[Nuseibeh93] B. Nuseibeh, A. Finkelstein e J. Kramer; *Fine-Grain Modelling*; 6th Annual Workshop on Software Reuse, 1993.

[Osterweil87] L. Osterweil; *Software Processes are Software Too*; 9th International Conference on Software Engineering, IEEE, 1987.

[REBOOT94] *REBOOT Newsletter*; no. 14, Setembro/1994.

[Rocha94] A.R.C. da Rocha e al; *Ambiente de Desenvolvimento de Software Baseado em Reutilização: Uma Proposta Técnica*; Programa de Engenharia de Sistemas e Computação, COPPE/UFRJ, 1994.

[Rocha96] A.R.C. da Rocha, C.M.L. Werner, G.H. Travassos e V.M.B. Werneck; *Processo de Desenvolvimento de Software Baseado em Reutilização*, Programa de Engenharia de Sistemas e Computação, COPPE/UFRJ, 1996.

[Silva92] J.L.M. da Silva; *Modelagem do Processode Desenvolvimento*; Programa de Engenharia de Sistemas e Computação, Tese de Mestrado, COPPE/UFRJ, Dezembro/1992.

[Whitenack95] B. Whitenack; *RAPPeL: A Requirements-Analysis-Process Pattern Language for Object Oriented Development*; em [Coplien95a].

[Yu94] E.S.K. Yu e J. Mylopoulos; *Understanding "Why" in Software Process Modeling, Analysis and Design*; 16th International Conference on Software Engineering, IEEE, Maio/1994.