DESENVOLVIMENTO DE SOFTWARE COM METODOLOGIA PRAXIS AUXILIADA PELA FERRAMENTA CASE RATIONAL ROSE

*André Fernando Tetto**
*Peter Jandl Junior***

RESUMO

Este trabalho pretende verificar como o emprego da metodologia Praxis pode ser suportado de maneira efetiva pelo uso conjunto da ferramenta CASE Rational Rose. A análise desta metodologia, de seus artefatos e da contribuição da ferramenta escolhida permite apresentar algumas das vantagens e desvantagens desta combinação no processo de desenvolvimento de software.

Palavras-chave: engenharia de software, processo de software, desenvolvimento de software, metodologia de desenvolvimento, CASE.

ABSTRACT

The aim of this study is to verify how the use of the Praxis methodology can be effectively supported by the use of the Rational Rose Case Tool. The analysis of this methodology, its artifacts and the contributions of the selected tool exposes some advantages and restrictions of the proposed association in the software development process.

Key words: software engineering, software process, software development, development methodology, CASE.

INTRODUÇÃO

Quando projetos de programas de computação complexos são realizados de maneira informal, isto é, sem auxílio de metodologias apropriadas de desenvolvimento, aumentam-se os riscos deste processo levando, com freqüência, a situações desagradáveis: cronogramas em atraso, insatisfação do cliente e baixa qualidade do software produzido. Quando funcionam é devido a talentos individuais, mas os sucessos quase nunca se repetem, sem considerar o desperdício de recursos e dinheiro quando assim conduzidos (PAULA, 2003; PRESSMAN, 2003; WIKIPEDIA, 2005).

A Engenharia de Software, disciplina que tem como uma de suas principais preocupações a oferta de metodologias e ferramentas apropriadas para o desenvolvimento de software, pode e deve ser utilizada para que os resultados esperados em um projeto de software possam ser alcançados (SOMMERVILLE, 2003).

Uma metodologia de desenvolvimento de software fornece os detalhes de "como fazer" para desenvolver um software envolvendo um amplo conjunto de tarefas que incluem: levantamento e análise dos requisitos, projeto, implementação e testes de software. Já as ferramentas proporcionam apoio automatizado ou semi-

* Analista de Sistemas (USF). Analista Programador Micro Mídia Informática.
** Mestre em Educação (USF) e Engenheiro de Eletrônica (UNICAMP). Professor de cursos de Graduação, Pós-graduação e Extensão da Universidade São Francisco e do Centro Universitário Padre Anchieta.

automatizado para as metodologias como, por exemplo, as ferramentas CASE (*Computer Aided Software Engineering*), que auxiliam nas diferentes fases do ciclo de vida do software através do uso de bases de dados e interfaces gráficas e textuais. Segundo Barrére (1999), uma ferramenta CASE tem como propósito auxiliar o desenvolvedor na maximização de suas habilidades intelectuais e criativas para a obtenção de software de mais alta qualidade com maior produtividade.

Este trabalho apresenta o estudo da metodologia de desenvolvimento de software Praxis, escolhida por ser uma metodologia brasileira, e sua interação com a ferramenta CASE Rational Rose (IBM, 2005A), muito utilizada nos âmbitos acadêmico e profissional, averiguando de que forma esta metodologia é suportada pela ferramenta escolhida, o que possibilita a determinação de algumas das vantagens e desvantagens desta combinação no desenvolvimento de software.

A METODOLOGIA DE DESENVOLVIMENTO PRAXIS

O Praxis (Processo para Aplicativos eXtensíveis InterativoS) é, com mais rigor, um processo de desenvolvimento de software que enfatiza o desenvolvimento de aplicações gráficas interativas, e foi idealizado pelo professor Wilson de Pádua Paula Filho (2003).

Inicialmente desenhado para suportar projetos didáticos em disciplinas de engenharia de software de cursos de informática e em programas de capacitação profissional em processo de software, pode também ser utilizado como base de treinamento preparatório para o uso dos processos RUP (*Rational Unified Process*), PSP (*Personal Software Process*), TSP (*Team Software Process*). Também é adequado ao desenvolvimento de projetos comerciais, desde que personalizado de acordo com a organização que o adotar (PAULA, 2003).

O processo Praxis abrange tanto métodos técnicos (requisitos, análise, desenho, testes e implementação) quanto métodos gerenciais (gestão de requisitos, de projetos e de configurações, além de garantia da qualidade). Propõe assim um ciclo de vida composto por fases que produzem um conjunto bem caracterizado de artefatos (documentos, modelos e relatórios) (PAULA, 2003). É baseado na tecnologia orientada a objetos, possuindo como notação de análise e desenho a UML (*Unified Modeling Language*) (FOWLEY; SCOTT, 2000; MATOS, 2003; IBM, 2005B).

Os fluxos do Praxis cobrem áreas chaves de processo do SW-CMM (*Software Capability Maturity Model*), um modelo de capacitação específico para a área de software (SEI/CMU, 2005), cujos padrões estão em conformidade com os propostos pelo IEEE (*Institute of Eletrical and Eletronic Engineers*) (IEEE, 1994) e reflete elementos do RUP (PAULA, 2002; PAULA, 2003).

Segundo Paula (2003), o uso, personalização e reprodução do processo Praxis é livre, desde que citadas suas fontes e também claramente identificadas as diferenças frente ao Praxis padrão, que garante para quem o utilizar o nível três (3) do SW-CMM (SEI/CMU, 2005), como destacado na Tabela 1.

Tabela 1. Praxis e o SW-CMM			
Número do Nível	Nome do Nível	Característica da Organização	Característica dos Processos
1	Inicial	Não segue rotinas	Processos caóticos
2	Repetitivo	Segue rotinas	Processos disciplinados
3	Definitivo	Escolhe rotinas	Processos padronizados
4	Gerido	Cria e aperfeiçoa rotinas	Processos previsíveis
5	Otimizante	Otimiza rotinas	Processos em melhorias contínuas

NOMENCLATURA

Descreve-se a seguir a nomenclatura empregada pelo Praxis, ilustrada na figura 1. Tal como o RUP, o Praxis abrange tanto fases quanto fluxos.

- **Fase**: divisão maior de um processo, para fins gerenciais, que corresponde aos pontos principais de aceitação por parte do cliente (divisões orientadas para gestão de projetos). Uma fase é composta por uma ou mais iterações.
- **Iteração**: divisões de uma fase, nas quais se atinge um conjunto bem definido de metas parciais de um projeto, é um exemplo de passo. Cada iteração possui um script.
- **Script**: conjunto de instruções que definem como uma iteração deve ser executada.
- **Fluxo**: subprocesso caracterizado por um tema técnico ou gerencial (divisões orientadas por disciplina de engenharia de software). Um fluxo é dividido em uma ou mais atividades.
- **Atividades**: passos constituintes de um fluxo.
- **Passos**: divisão formal de um processo, com pré-requisitos, entradas, critérios de aprovação e resultados definidos.

* Analista de Sistemas (USF). Analista Programador Micro Mídia Informática.
** Mestre em Educação (USF) e Engenheiro de Eletrônica (UNICAMP). Professor de cursos de Graduação, Pós-graduação e Extensão da Universidade São Francisco e do Centro Universitário Padre Anchieta.

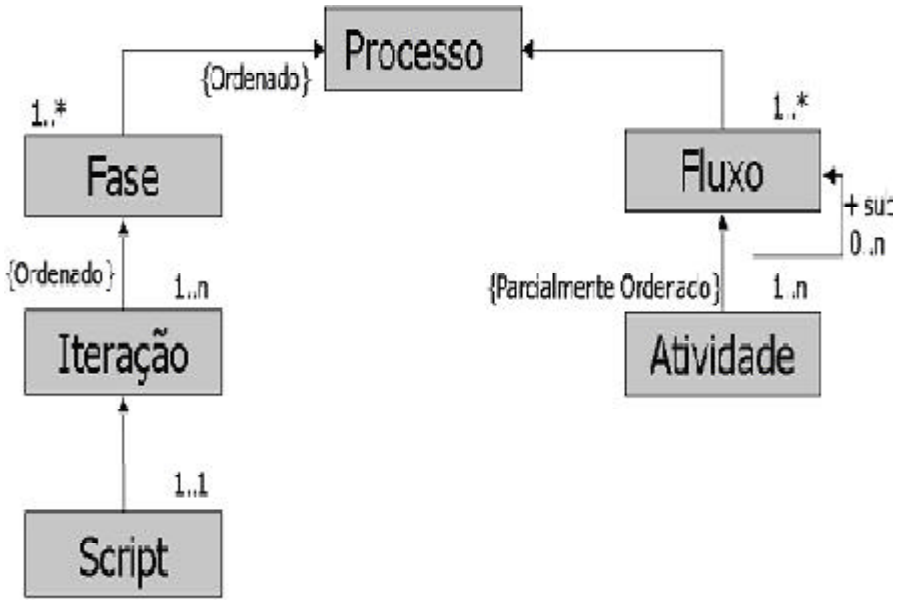

Figura 1 - Estrutura de Processo, Fase e Fluxo.

As relações existentes entre fases, iterações, scripts, fluxos, atividades e artefatos no processo Praxis descrevem o funcionamento do processo.

FASES

A divisão das fases obedece ao modelo de ciclo de vida de entrega evolutiva, sendo que o término de cada fase é sempre determinado pela entrega e aprovação de um conjunto preestabelecido de artefatos (resultados) (PAULA, 2003). Como no RUP, o Praxis apresenta as seguintes fases:

- **Concepção**: as necessidades dos usuários e os conceitos da aplicação são analisados o suficiente para justificar a especificação de um produto de software.
- **Elaboração**: a especificação do produto é detalhada o suficiente para modelar conceitualmente o domínio do problema, validar os requisitos em termos desse modelo conceitual e permitir um planejamento detalhado da fase de construção.
- **Construção**: é desenvolvida, ou seja, desenhada, implementada e testada uma versão completamente operacional do produto, a qual atende aos requisitos especificados.

- **Transição**: passagem do produto do ambiente de desenvolvimento para o ambiente do usuário.

ITERAÇÕES

Uma iteração é o resultado da divisão de uma fase. As fases são divididas assim (PAULA, 2003):

Concepção

- Ativação: levantamento e análise das necessidades dos usuários e conceitos da aplicação, em nível de detalhe suficiente para justificar a especificação de um produto de software.

Elaboração

- Levantamento de requisitos: levantamento das funções, interfaces e requisitos não-funcionais desejados para o produto.
- Análise dos requisitos: modelagem conceitual dos elementos relevantes do domínio do problema e uso desse modelo para validação dos requisitos e planejamento detalhado da fase de construção.

Construção

- Desenho implementável: definições internas e externas dos componentes de um produto de software, em nível suficiente para decidir as principais questões de arquitetura e tecnologia e também permitir o planejamento da fase de construção.
- Liberação: implementação de um subconjunto de funções do produto que será avaliado pelos usuários; após a implementação de todas as liberações, o produto estará totalmente implementado.
- Testes Alfa: realização dos testes de aceitação, no ambiente de desenvolvimento, juntamente com elaboração da documentação do usuário e possíveis planos de Transição.

Transição

- Testes Beta: realização dos testes de aceitação no ambiente dos usuários.
- Operação piloto: operação experimental do produto em instalação piloto do cliente, com a resolução de eventuais problemas através de processo de manutenção.

SCRIPTS

Para cada iteração o *script* indica os artefatos que consome (insumos) e produz (resultados), os respectivos critérios de entrada (pré-requisitos) e de saída (critérios de aprovação) e um conjunto de atividades sugeridas. Toda iteração possui como único pré-requisito o término da iteração anterior e todos os insumos de uma iteração são os resultados da iteração anterior. Por essa razão os pré-requisitos e os insumos não são detalhados nos *scripts* das iterações do Praxis (PAULA, 2003).

FLUXOS

No Praxis, os fluxos podem ser de natureza técnica ou gerencial. Os fluxos de natureza técnica são: Requisitos, Análise, Desenho, Implementação, Testes e Engenharia de Sistemas; os fluxos de natureza gerencial são: Gestão de Projetos, Gestão da Qualidade e Engenharia de Processos. Os fluxos definem os papéis desempenhados pelos participantes dos projetos e algumas atividades requerem a participação de vários papéis em sua execução (PAULA, 2003).

ARTEFATOS

Os resultados produzidos e os insumos consumidos nos passos do Praxis são chamados de artefatos do processo, cujos tipos são:

- Documento: artefato produzido por ferramenta de processamento de texto ou hipertexto, para fins de documentação dos principais aspectos de engenharia de um projeto, incluindo aspectos selecionados dos modelos e aspectos não modeláveis.
- Modelo: artefato de uma ferramenta técnica específica, produzido e utilizado nas atividades de um dos fluxos do processo.
- Relatório: artefato que relata as conclusões das atividades do projeto.

DOCUMENTOS

Este processo utiliza-se de vários documentos diferentes que detalham aspectos específicos do projeto:

- **PESw** (Proposta de Especificação do Software): delimita preliminarmente o escopo de um projeto, contendo um plano da fase de elaboração.
- **ERSw** (Especificação dos Requisitos do Software): descreve o conjunto de requisitos especificados para um produto de software.
- **PDSw** (Plano de Desenvolvimento do Software): descreve os compromissos que o fornecedor assume em relação ao projeto quanto a recursos, custos, riscos e outros aspectos gerenciais.
- **PQSw** (Plano de Qualidade do Software): descreve os procedimentos de garantia da qualidade que serão adotados no projeto.
- **DDSw** (Descrição do Desenho do Software): descreve os aspectos mais importantes no desenho do software.
- **DTSw** (Descrição dos Testes do Software): descreve os planos e as especificações dos testes que serão executados.
- **MUSw** (Manual do Usuário do Software): serve como referência para uso do produto.

Os únicos documentos gerenciais são o **PDSw** e **PQSw**, sendo os demais considerados documentos técnicos. Esses documentos são tipicamente produzidos através de uma ferramenta de edição de textos (PAULA, 2003).

MODELOS

Seguem os modelos utilizados pelos Praxis e as ferramentas necessárias para sua confecção:

- **CRSw** (Cadastro dos Requisitos do Software): contém os requisitos levantados, assim como referências aos itens correspondentes dos modelos seguintes. [Planilha, banco de dados].
- **MASw** (Modelo de Análise do Software): contém os conceitos do domínio do problema a resolver que sejam relevantes para a validação dos requisitos. [Ferramenta de modelagem orientada a objetos].
- **MPPSw** (Memória de Planejamento do Projeto do Software): contém a informação necessária para o acompanhamento de tamanhos, esforços, custos, prazos e riscos dos projetos. [Planilha, ferramenta de gestão de projetos].
- **MDSw** (Modelo de Desenho do Software): detalha a estrutura lógica e física do produto, em termos de seus componentes. [Ferramenta de modelagem orientada a objetos].
- **BTRSw** (Bateria de Testes de Regressão do Software): conjunto dos *scripts* dos testes de regressão. [Ferramenta de desenvolvimento, ferramenta de testes].
- **CFSw** (Códigos Fontes do Software): conjunto dos códigos fontes produzidos. [Ferramenta de desenvolvimento].
- **CESw** (Códigos Executáveis do Software): conjunto dos códigos executáveis produzidos. [Ferramenta de desenvolvimento].

Segundo Paula (2003), a **MPPSw** é o único modelo gerencial.

RELATÓRIOS

Existem vários relatórios, que são produzidos por diferentes responsáveis:

- **RTSw** (Relatórios dos Testes de Software): descreve os resultados dos testes realizados. [Grupo de testes do projeto].
- **RRSw** (Relatórios de Revisão do Software): descreve as conclusões da revisão de um artefato. [Grupo revisor do artefato].
- **RISw** (Relatórios de Inspeção do Software): descreve as conclusões da inspeção de um artefato. [Grupo inspetor de artefato].
- **RAQSw** (Relatórios das Auditorias da Qualidade do Software): descreve as conclusões de uma auditoria da qualidade realizada. [Grupo de garantia da Qualidade].
- **RAPSw** (Relatórios de Acompanhamento do Projeto do Software): descreve esforços, custos, prazos e riscos até a data corrente. [Gerente de projeto].
- **RFPSw** (Relatório Final do Projeto de Software): relatório de balanço final do projeto. [Gerente do projeto].

Os três primeiros são de caráter técnico e os demais de caráter gerencial. O plano da qualidade prevê as datas de emissão dos relatórios de testes, revisões e

auditorias. Os relatórios de acompanhamentos são produzidos com a periodicidade especificada no plano de desenvolvimento, geralmente por iteração (PAULA, 2003).

CRONOGRAMA DOS ARTEFATOS

As tabelas 2 e 3 mostram o relacionamento entre as iterações, os modelos e os documentos do Praxis (PAULA, 2003).

	PESw	ERSw	PDSw	PQSw	DDSw	DTSw	MUSw
Ativação	C						
Levantamento dos requisitos		P	P				
Análise dos requisitos		C	C	C	P		
Desenho implementável		A	A	A	I	P	
Liberação		A	A	A	I	I	
Testes Alfa		A	A	A	C	C	P
Testes Beta		A	A	A	A	A	A
Operação piloto		A	A	A	A	A	C
P - Artefato começa a ser produzido							
I - Versão incompleta							
C - Artefato completado							
A - Artefato pode ser alterado							

Tabela 2. Relação entre iterações e documentos

	CRSw	MASw	MPPSw	MDSw	BTRSw	CFSw	CESw
Ativação							
Levantamento dos requisitos	P	P	P				
Análise dos requisitos	C	C	C	P			
Desenho implementável	A	A	A	I	P	P	P
Liberação	A	A	AI	I	I	I	I
Testes Alfa	A	A	A	C	C	C	C
Testes Beta	A	A	A	A	A	A	A
Operação piloto	A	A	A	A	A	A	A
P - Artefato começa a ser produzido							
I - Versão incompleta							
C - Artefato completado							
A - Artefato pode ser alterado							
Tabela 3. Relação entre iterações e modelos							

GARANTIA DE QUALIDADE

Procedimentos de controle são executados de maneira uniforme, em diferentes iterações do ciclo de vida, sendo sua conclusão condição necessária para que as iterações do projeto sejam consideradas aprovadas, passando-se às iterações seguintes. Algumas iterações requerem aprovação dos usuários chave para determinar se os requisitos foram corretamente interpretados pelos desenvolvedores, ou do cliente quando envolvem decisões de continuidade do projeto (fim da Concepção e Elaboração) ou aceitação do produto (fim da Construção e Transição). Esses pontos de aceitação pelo cliente demarcam, por definição, os finais das fases (PAULA, 2003).

VANTAGENS E DESVANTAGENS

Como vantagens pode-se indicar que o Praxis é baseado na tecnologia orientada a objetos; possui como notação de análise e desenho a UML; seus fluxos cobrem áreas chaves de processo do SW-CMM, garantindo inicialmente o nível 3 (três); seus padrões são conformes aos padrões de engenharia de software do IEEE; reflete elementos do Processo Unificado; e é um processo iterativo que pode

ser utilizado para fins didáticos e comerciais desde que personalizado (PAULA, 2003).

Como desvantagens tem-se que este processo possui uma comunidade pequena de usuários, talvez devido ao pequeno período de sua divulgação, e que ainda foram relatados poucos casos de sucesso do seu uso (ÁLVARES, 2000; CARVALHO, 2000; BORGES, 2002; PERES, 2002; BORGES; PAULA, 2003; SANTOS, 2004). Também não pode ser utilizado ou considerado como uma metodologia ágil, pois exige muita documentação.

AVALIAÇÃO COMPARATIVA ENTRE PRAXIS E RUP

Embora não constitua um dos objetivos deste trabalho, dada a similaridade de nomenclatura de alguns elementos do Praxis com relação ao RUP, mais conhecido nos meios acadêmico e profissional, é conveniente a realização de uma análise comparativa mínima para estabelecer outras semelhanças e caracterizar diferenças entre estes processos.

Segundo Kruchten (2003), o RUP é simultaneamente um processo de engenharia de software (pois fornece uma abordagem disciplinada para assumir tarefas e responsabilidades dentro de uma organização de desenvolvimento), um produto de processo (pois é desenvolvido e mantido pela Rational Software e integrado com seu conjunto de ferramentas de desenvolvimento de software) e uma estrutura de processo (pois pode ser adaptada e estendida para compor as necessidades de uma organização que o adote). Seu objetivo é assegurar a produção de software de alta qualidade que satisfaça as necessidades de seus usuários finais dentro de prazos e orçamentos previsíveis.

Embora tanto o Praxis quanto o RUP (IBM, 2005C) sejam processos de software baseados na tecnologia orientada a objetos, no Processo Unificado e que utilizam a notação UML, possuem objetivos diferentes. Enquanto o RUP destina-se claramente ao desenvolvimento de aplicações comerciais complexas, o objetivo do Praxis é o desenvolvimento de projetos didáticos em disciplinas de engenharia de software de cursos de informática e em programas de capacitação profissional. Desta forma o Praxis não possui como objetivo substituir ou mesmo concorrer com o RUP.

O Praxis não detalha papéis, deixando esta tarefa a cargo das organizações que o adotarem; já no RUP o conceito central do processo está em um trabalhador, isto é, em uma posição (IBM, 2005C; WIKIPEDIA, 2005). Finalmente, outra diferença expressiva é que o RUP é um produto de processo, ou seja, para ser utilizado precisa ser comprado, enquanto o Praxis é um processo de uso, reprodução e personalização livre.

AVALIAÇÃO DO SUPORTE NA FERRAMENTA RATIONAL ROSE AO PROCESSO PRAXIS

Segundo Matos (2003), o Rational Rose é uma ferramenta CASE para UML que implementa as recomendações da OMG (*Object Management Group*) e que é baseada no conceito de modelo de negócios. É uma ferramenta robusta, com o

propósito de desenvolver soluções que atendam às necessidades do negócio do cliente, que vão desde uma simples solução localizada até soluções complexas baseadas em ambientes distribuídos. A construção de um modelo de negócios envolve diagramas de casos de uso, diagramas de objetos, diagramas de classes e demais diagramas da UML, os quais são suportados pelo Rational Rose. Em sua versão mais completa oferece suporte bastante amplo as linguagens de programação existentes, entre elas C++, Visual Basic e Java.

A fim de se avaliar o suporte efetivo oferecido pelo Rational Rose quando se utiliza o processo Praxis, foi necessário o desenvolvimento de um projeto de software. Como plataforma de desenvolvimento escolheu-se o Java devido a sua relevância atual (JANDL, 2002; SUN, 2004). O Rational Rose foi usado na confecção parcial ou total de um conjunto dos artefatos mais importantes do processo.

O projeto de software desenvolvido foi o Funny 1.0, um sistema de gestão de contas a pagar e receber para uma loja fictícia de equipamentos eletrônicos denominada Circuito Integrado. A principal motivação para a escolha desse projeto foi a existência de conhecimento prévio sobre a lógica do negócio desse tipo de sistema. Esse sistema possibilita a inclusão, consulta, alteração, exclusão, liquidação (pagamento) e estorno da liquidação (cancelamento do pagamento) de títulos a pagar e a receber. Possibilita também a importação de títulos a pagar e a receber do sistema de compras e vendas já existente da loja e também a inclusão, consulta, alteração e exclusão dos responsáveis desses títulos (clientes quando títulos cadastrados como contas a receber; fornecedores quando títulos cadastrados como contas a pagar).

Neste trabalho forma utilizados: processo Praxis versão 2.0, Rational Rose 98 Enterprise Edition (com suporte para versão JDK 1.1.X da plataforma Java) e também o Borland JBuilder X.

O critério de seleção de artefatos do processo Praxis escolhidos para avaliar o suporte do Rational Rose considerou a importância dos artefatos no processo e se sua confecção permitiria a avaliação de suporte. Embora tenha se procurado um conjunto mínimo, para evitar a descaracterização do processo, foram incluídos os artefatos **PESw** e **MPPSw**, cuja confecção não necessita uma ferramenta CASE. Sendo assim o conjunto de artefatos escolhidos e desenvolvidos foi: documentos (**PESw**, **ERSw** e **DDSw**) e modelos (**MPPSw**, **MASw**, **MDSw** e **CFSw**). Estes artefatos foram total ou parcialmente produzidos através do Rational Rose.

O desenvolvimento dos artefatos seguiu o cronograma definido pelo processo Praxis (veja tabelas 2 e 3). Também foi realizado um controle do tempo necessário para a confecção de cada artefato (Tabela 4).

Os primeiros artefatos produzidos e avaliados foram o **ERSw** e **MASw**, desenvolvidos em paralelo na fase de elaboração. Para o **ERSw** o Rational Rose foi utilizado para a confecção dos diagramas de contexto, de casos de uso e de classes persistentes. Para o **MASw** o Rose permitiu a confecção total do modelo, o que inclui todos os diagramas existentes na **ERSw**, além do diagrama de classes. O

processo Praxis recomenda o uso da notação UML para a confecção desses diagramas. Especificamente em relação ao **MASw**, existe a recomendação para que sejam desenvolvidas classes de três tipos: Fronteira (*boundary*), que modelam as interfaces do produto com os usuários e com outros sistemas; Entidades (*entity*), que modelam informações persistentes; e Controle (*control*), que coordenam o fluxo de um caso de uso complexo, encapsulando lógica que não se enquadra naturalmente nas responsabilidades das entidades.

Em particular, o projeto desenvolvido neste trabalho possui apenas os tipos de classes de fronteira e de entidade. As classes de controle não foram necessárias, pois não havia casos de uso complexos a serem coordenados por este tipo de classe, podendo ser coordenados pelas classes de fronteira sem infringir as recomendações do processo Praxis, que menciona que casos de uso simples podem ser coordenados por classes de fronteira. O Rose suportou adequadamente todos estes desenhos, de acordo com o recomendado pelo processo Praxis, tal como indicado na tabela 4.

Ainda na fase de construção foram desenvolvidos em paralelo os artefatos **DDSw** e **MDSw**, como definido pelo processo Praxis. É importante mencionar que foram feitas apenas as iterações Desenho Implementável e Liberação 1: Manutenção de Títulos, pois através do desenvolvimento dessas iterações foi possível coletar material suficiente para o objetivo deste trabalho.

Para a **DDSw** foi possível utilizar o Rose para a confecção dos diagramas da estrutura dinâmica do produto, diagramas de visão lógica (pacotes lógicos, classes, realizações dos casos de uso e interação), e diagramas de visão física (componentes físicos). Já para o **MDSw** o Rose permitiu a confecção total do modelo, o que inclui todos os diagramas existentes na **DDSw**. Como antes, existe a recomendação de emprego da notação UML na confecção dos diagramas, ou seja, uso dos diagramas de colaboração para descrever a estrutura dinâmica do software e dos diagramas de pacotes para representar a visão lógica do sistema em termos de camadas e componentes.

O Rose suportou adequadamente a construção destes diagramas, com pequenas restrições nos diagramas de casos de uso e de interações devido à versão utilizada. Não foi possível a criação do desenho das tabelas de banco de dados utilizadas pelo projeto, mas apenas a geração de *scripts* de criação de base de dados a partir das classes (no caso das classes de entidade), e executá-los em um gerenciador de banco de dados. Os tempos consumidos na confecção da **DDSw** e do **MDSw** estão indicados na tabela 4.

O **CFSw** foi o último artefato desenvolvido e analisado neste trabalho. O Rose gerou adequadamente a codificação das classes na plataforma Java de acordo com o diagrama de classes do **MDSw**. O código gerado consiste nos atributos e nas assinaturas dos métodos, incluindo os construtores destas classes. Após a geração do código foi possível sua edição através do Borland JBuilder X, indicando que outros ambientes de programação poderiam ser empregados.

Uma análise das classes geradas permitiu verificar que não houve a inclusão de código desnecessário, exceto a declaração de algumas classes que realizam associações no diagrama de classes. Nesta etapa o Rose não realiza uma compilação das classes em busca de erros. Também não existe a necessidade de um desenho prévio dos componentes do software antes de geração do código, pois caso não existam, esses desenhos serão gerados automaticamente. A documentação das classes geradas também é produzida automaticamente.

Estudando-se a característica do Rose de engenharia reversa da codificação das classes foi notada a realização de uma compilação com o propósito de se encontrarem erros de sintaxe e de configuração. Durante a realização da engenharia reversa também foi criado o desenho dos componentes de software referentes às classes do software. O Rose não suportou a engenharia reversa das classes contendo o desenho das interfaces de usuário (classes de fronteira) produzidas pela ferramenta de programação (no caso o JBuilder X), conforme recomendação do processo Praxis. Sendo assim todas as classes contendo interfaces de usuário foram desenhadas totalmente no Rose, situação que também possibilitou a geração automática de sua documentação.

Finalmente o **MPPSw** foi utilizado para armazenar o tempo necessário para a confecção de cada artefato, tal como mostra a tabela 4.

Artefato	Confecção	Tempo	Suporte
PESw	Não aplicável		Não avaliado
ERSw	Total	19h45m	Adequado
DDSw	Total com restrições	16h00m	Adequado
MPPSw	Não aplicável		Não avaliado
MASw	Total	15h30m	Adequado
MDSw	Total com restrições	10h00m	Adequado
CFSw	Total	06h00m	Adequado

Tabela 4. Tempos de produção dos artefatos e adequação do suporte do Rational Rose

CONCLUSÃO

No geral o suporte da ferramenta CASE Rational Rose oferecido ao processo Praxis foi bastante satisfatório, pois dos onze diagramas recomendados nenhum problema foi encontrado em sete deles, existindo pequenas restrições na confecção dos demais. Contudo, o fato da engenharia reversa não ser possível para classes contendo as interfaces de usuário desenhadas no JBuilder foi um aspecto

ruim. Isto exigiu a separação das fontes das classes de interface de usuário geradas pelo Rose e pelo JBuilder.

Os tempos mensurados, embora razoáveis, são apenas referenciais e provavelmente podem ser reduzidos com um maior domínio da ferramenta CASE a ser obtido através de seu uso continuado.

Constatou-se assim que projetos de software de cunho acadêmico que utilizem o processo Praxis podem se beneficiar do apoio oferecido pelo Rational Rose, principalmente se for possível o uso de versões mais recentes, que incluem suporte mais adequado para a notação UML, além de novas características.

Outra contribuição é a verificação que projetos comerciais de software também podem utilizar o Praxis combinado com o Rational Rose para o seu desenvolvimento. No entanto é recomendável a personalização do processo de acordo com as necessidades da organização.

Este trabalho poderia ser continuado de diversas formas, entre elas: desenvolvendo-se outras customizações e extensões para o Praxis, tais como nos trabalhos de Álvares (2000) e Peres (2002); ou efetuando-se avaliações semelhantes do suporte de outras ferramentas CASE em relação ao processo Praxis padrão ou de suas extensões.

REFERÊNCIAS BIBLIOGRÁFICAS

ÁLVARES, Patrícia M. R. de S. *WebPraxis - Um Processo Personalizado para Projetos de Desenvolvimento para a Web*. In Anais da IV Semana de Pós-Graduação em Ciência da Computação (SPG'2000), 28/08/00 a 30/08/00, Universidade Federal de Minas Gerais, Belo Horizonte. *On-Line*: http://www.dcc.ufmg.br/pos/html/spg2000/anais/pmarques/pmarques.htm, recuperado em 20/04/2005.

BARRÉRE, Tathiana da Silva; PRADO, Antonio Francisco; BONAFE, Vitor César. *CASE Orientada a Objetos com Múltiplas Visões e Implementação Automática de Sistemas – MVCASE*. São Carlos, 1999. On-Line: http://www.inf.ufsc.br/sbes99/anais/SBES-Completo/11.pdf, recuperado em 25/03/2004.

BORGES, Eduardo P. *Uma Ferramenta de Apoio à Gestão de Métricas para o Praxis*. In Anais da VI Semana de Pós-Graduação em Ciência da Computação (SPG'2002), 04/09/02 a 06/09/02, Universidade Federal de Minas Gerais, Belo Horizonte. *On-Line*: http://www.dcc.ufmg.br/pos/html/spg2002/anais/eborges/eborges.htm, recuperado em 20/04/2005.

BORGES, Eduardo O.; PAULA FILHO, Wilson de Pádua. *Um Modelo de Medição para Processos de Desenvolvimento de Software* In Anais do V Simpósio Internacional de Melhoria de Processo de Software (SIMPROS 2003), 03/11/03 a 05/11/03, Recife. *On-Line*: http://www.simpros.com.br/simpros2003/upload/arquivos_PDF/Artigos/ART_09.pdf, recuperado em 20/04/2005.

CARVALHO, Júnia G. *Praxis Mentor - Uma Ferramenta de Apoio à Utilização de um Processo de Desenvolvimento de Software*. In Anais da IV Semana de Pós-Graduação em Ciência da Computação (SPG'2000), 28/08/00 a 30/08/00, Universidade Federal de Minas Gerais, Belo Horizonte. *On-Line*: http://www.dcc.ufmg.br/pos/html/spg2000/anais/junia/junia.htm, recuperado em 20/04/2005.

FOWLEY, Martin; SCOTT, Kendall. *UML Essencial*. 2.ed. Porto Alegre: Bookman, 2000.

IBM. *RATIONAL Software*. On-Line: http://www-306.ibm.com/software/rational/, recuperado em 20/04/2005 (A).

IBM. *UML Resource Center*. On-Line: http://www-306.ibm.com/software/rational/uml/, recuperado em 20/04/2005 (B).

IBM. *Rational Unified Process*. On-Line: http://www-306.ibm.com/software/awdtools/rup/, recuperado em 20/04/2005 (C).

IEEE. IEEE Standards Collection – Software Engineering. IEEE, New York – NY, 1994.

JANDL, Peter Junior. *Introdução ao Java*. São Paulo: Berkeley, 2002.

KRUCHTEN, Philippe. *Introdução ao RUP: Rational Unified Process*. Rio de Janeiro: Ciência Moderna, 2003.

MATOS, Alexandre Veloso de. *UML: Prático e Descomplicado*. 3.ed. São Paulo: Érica, 2003.

PAULA FILHO, Wilson de Pádua. *PRAXIS 2.0*. On-Line: http://www.wppf.uaivip.com.br/praxis, recuperado em 08/05/2004.

PAULA, Wilson de Pádua, Filho. *Engenharia de Software: Fundamentos, Métodos e Padrões*. 2.ed. Rio de Janeiro: LTC, 2003.

PERES JR, Valdo N. *Uma Extensão do Praxis para a Arquitetura J2EE*. In Anais da VI Semana de Pós-Graduação em Ciência da Computação (SPG'2002), 04/09/02 a 06/09/02, Universidade Federal de Minas Gerais, Belo Horizonte. *On-Line*: http://www.dcc.ufmg.br/pos/html/spg2002/anais/valdo/valdo.htm, recuperado em 20/04/2005.

PRESSMAN, Roger S. *Engenharia de Software*. 5.ed. São Paulo: MCGRAW-HILL, 2003.

SEI/CMU. *Capability Maturity Model for Software (SW-CMM)*. Software Engineering Institute. Carnegie Mellon University. On-Line: http://www.sei.cmu.edu/cmm/cmm.html. Recuperado em 06/12/2004.

SANTOS, Daniela A. S. dos. *Personalização e Implantação de Procedimentos de Gestão da Qualidade dentro de um Programa de Melhoria*. In Anais do III Simpósio Brasileiro de Qualidade de Software (SBQS2004), Universidade Católica de Brasília, 31/05/04 a 04/06/04, Brasília. *On-Line*: http://www.mct.gov.br/Temas/info/Dsi/PBQP/III SBQS/ST1_4.pdf, recuperado em 20/04/2005.

SOMMERVILLE, Ian. *Engenharia de Software*. 6.ed. São Paulo: Pearson Education, 2003.

SUN. *Java Technology*. On:Line: http://java.sun.com/, recuperado em 30/07/2004.
WIKIPEDIA. *Rational Unified Process*. On-Line: http://en.wikipedia.org/wiki/Rational_Unified_Process, recuperado em 20/04/2005.