Iniciação Científica **CESUMAR**
Junho 2006, v. 08, n.01, p.27-33 - Edição Especial

DESENVOLVIMENTO DE *SOFTWARE*: USO DA PROTOTIPAÇÃO NA FASE DE CONCEPÇÃO DO MODELO DE PROCESSO DE SOFTWARE RUP

Douglas Eduardo Andreto[1], Ana Carolina Romanini Gonçalves Vicente[1],
Alexandre Palota Siviero[1], Aline Maria Malachini Miotto[2],
Fabrício Ricardo Lazilha[2], Márcia Cristina Dadalto Pascutti[2],
Munif Gebara Junior[2], Robinson Patroni[2]

RESUMO: Nos dias atuais, produzir *software* com qualidade e eficiência deixou de ser uma característica desejável e passou a ser necessidade para um produto de *software*. Nota-se que o uso de processos de desenvolvimento de *softwares* formalizados garante ao *software* qualidade e eficiência durante todo o ciclo de vida da produção de um sistema. Este trabalho tem como finalidade estudar e aplicar uma abordagem formalizada para o desenvolvimento do *software*, neste caso o RUP (*Rational Unified Process)*, destacando suas vantagens e principais características.

PALAVRAS-CHAVE: Desenvolvimento de software, RUP (*Rational Unified Process*); prototipação.

SOFTWARE DEVELOPMENT: USE OF PROTOTYPING IN THE CONCEPTION STAGE OF RUP SOFTWARE PROCESS MODEL

ABSTRACT: Nowadays, to produce software with quality and efficiency is not just a desirable characteristic, but also a necessity for a software product. The use of formalized software development processes guarantee quality and efficiency to the software during the whole production life cycle of a system. This work has aimed at studying and applying a formalized approach for software development, in this case, the RUP (Rational Unified Process), highlighting its advantages and main characteristics.

KEYWORDS: Software Development, RUP (Rational Unified Process); Prototyping.

1 Acadêmicos do Curso de Processamento de Dados do CESUMAR – Centro Universitário de Maringá
2 Docentes do Curso de Processamento de Dados do CESUMAR – Centro Universitário de Maringá

1. INTRODUÇÃO

Nos últimos anos, o uso da tecnologia vem ganhando um importante papel nos processos de tomada de decisão das empresas. Cada vez mais as organizações precisam de agilidade e confiabilidade em seus processos. Dentro deste contexto, os aplicativos computacionais se apresentam como uma boa alternativa, pois permitem a disponibilização e o processamento eficiente de uma grande quantidade de informações.

Não obstante, os benefícios obtidos com a tecnologia dependem fundamentalmente da qualidade desta. Este trabalho tem como objetivo principal investigar e experimentar, por meio do desenvolvimento de uma aplicação, alternativas para o desenvolvimento de softwares eficientes e de qualidade. Um dos principais recursos que vêm sendo discutidos e utilizados amplamente para a garantia da qualidade de um *software* é o uso de um processo de desenvolvimento de *software* formalizado. Um processo de *software* é um conjunto de atividades e resultados associados que geram um produto de *software*. Este processo pode envolver o desenvolvimento de *softwares* desde o início, embora cada vez mais ocorra o caso de um *software* novo ser desenvolvido mediante a expansão e a modificação de sistemas já existentes (SOMMERVILLE, 2003).

Vários modelos de processo de *software* foram propostos na literatura, tais como: RUP (*Rational Unified Process*) (KRUCHTEN, 2003), Catalysis (D'SOUZA *et al*., 1999) e XP (*Extreming Programing*) (BECK 1999). Esses modelos apresentam características que os tornam mais adequados à construção de determinados tipos de aplicação. Neste trabalho optou-se por utilizar o RUP, uma vez que este modelo vem sendo amplamente utilizado em projetos de desenvolvimento de *softwares* orientados a objetos. Este modelo também prevê a construção de protótipos para refinamento de requisitos na fase de concepção, característica esta desejada neste trabalho. Dentro deste contexto, este trabalho tem como objetivo demonstrar o uso do modelo RUP, em sua fase inicial de concepção, para o desenvolvimento de um *sistema para comunicação corporativa*.

Dessa forma, este trabalho apresenta na seção 2 uma descrição detalhada sobre *análise orientada a objetos*, uma vez que a abordagem utilizada para modelagem do sistema se baseia neste paradigma. A seção 3 apresenta a UML (*Unified Modeling Language)* (BOOCH *et al.,* 2000) - metodologia utilizada para modelagem e diagramação do sistema proposto neste trabalho. Na seção 4 é apresentada uma discussão sobre o modelo RUP, seus processos e suas características. A seção 5 discute as principais características do sistema em desenvolvimento neste trabalho, bem como algumas decisões de projeto que foram tomadas antes do início da modelagem deste sistema. Na seção 6 apresentam-se os resultados obtidos até o presente momento com o desenvolvimento deste trabalho. Finalmente, na seção 7 são apresentadas as considerações finais desta pesquisa.

2. ANÁLISE ORIENTADA A OBJETOS

A análise orientada a objetos traz uma forma diferente de modelar e desenvolver sistemas, onde dados e processos são organizados e manipulados por objetos, e não por programas.

De acordo com Pressman (2005), a orientação a objetos baseia-se em conceitos da realidade – objetos, estados e estímulos – para desenvolver e modelar sistemas. Esses conceitos tornam natural o uso da orientação a objetos, em que tais conceitos são os mesmos que já conhecemos, porém aplicados ao desenvolvimento de sistemas.

A análise orientada a objetos tem como objetivo desenvolver modelos de análise que satisfaçam um conjunto de requisitos definidos pelo cliente, que descrevam as informações, as funções e o comportamento de seus elementos constituintes. A análise orientada a objetos define o mundo como uma população de objetos interativos – cada um sendo um conjunto de dados e funcionalidades – com estrutura de dados, comportamentos e eventos que mudam esses comportamentos ou suas operações. (MARÇULA; BENINI FILHO, 2005)

Enquanto no modelo estruturado se focaliza o programa, que possui processos que realizam funções sobre os dados, no modelo orientado a objetos focaliza-se o objeto, ou melhor, as classes de objetos existentes no sistema. Essas classes possuem atributos (que correspondem aos dados) e operações (que correspondem aos processos).

De acordo com Silva (2001), as vantagens da análise orientada a objetos são:

* reutilização dos objetos: os dados são manipulados por objetos, não ficando engessados dentro de programas, isto é, os objetos construídos podem ser utilizados por diferentes sistemas;
* modularidade: o sistema é formado por objetos e não por programas, facilitando o trabalho e a manutenção desses objetos no futuro;
* utilização dos mesmos conceitos da realidade (objetos, estados e estímulos) para modelagem e construção de sistemas de *software*.

Para produzir e representar a modelagem do sistema na análise orientada a objetos pode-se utilizar a linguagem padronizada UML. Dentro deste contexto, na próxima seção são apresentados alguns dos principais conceitos de UML.

3. UML(*Unified Modeling Language*)

Segundo Booch *et al.* (2000)**,** UML é uma linguagem-padrão para a elaboração da estrutura de projetos de *software*, podendo ser empregada para a visualização, a especificação, a construção e a documentação de artefatos de *software*. A UML somente faz parte de um método de desenvolvimento de *softwares*, sendo independente de processo; ou seja, pode ser utilizada por diferentes modelos de processo de *software*, tais como: Catalysis, RUP, Cascata, dentre outros. Sua abrangência vai desde sistemas de informações corporativos até aplicações baseadas em *Web* e sistemas embutidos de tempo real.

A UML não é uma linguagem de programação visual, e sim, uma linguagem para modelagem visual. Mesmo oferecendo suporte a diversas linguagens de programação, a UML é apenas

uma linguagem para a construção de artefatos de *software* (BOOCH *et al.,* 2001)

Segundo Melo (2004), a UML na versão 2.0 é composta de 13 diagramas. São eles:

* diagrama de classes: apresenta elementos conectados por relacionamentos e é usado para exibir entidades do mundo real, além de elementos de análise e projeto;
* diagrama de objetos: apresenta objetos e valores de dados e corresponde a uma instância do diagrama de classes, mostrando o estado de um sistema em um determinado ponto do tempo;
* diagrama de componentes: mostra as dependências entre componentes de *software*, apresentando suas interfaces;
* diagrama de implantação: mostra a arquitetura do sistema em tempo de execução, as plataformas de *hardware*, artefatos de *software* e ambientes de *software* (como sistemas operacionais e máquinas virtuais);
* diagrama de pacotes: usado para organizar elementos de modelo e mostrar dependências entre eles;
* diagrama de estrutura composta: usado para mostrar a composição de uma estrutura; é útil em estruturas compostas de estruturas complexas ou em projetos baseados em componentes;
* diagrama de casos de uso: mostra os casos de uso, atores e seus relacionamentos que expressam a funcionalidade de um sistema;
* Diagrama de visão geral: uma variação do diagrama de atividades que mostra de uma forma geral o fluxo de controle dentro de um sistema ou processo de negócios; cada nó ou atividade dentro do diagrama pode representar outro diagrama de interação.
* diagrama de seqüências: mostra as interações que correspondem a um conjunto de mensagens trocadas entre os objetos e a ordem em que essas mensagens acontecem;
* diagrama temporal: mostra a mudança de estado de um objeto numa passagem de tempo, em resposta a eventos externos;
* diagrama de comunicação: é um antigo diagrama de colaboração, que mostra objetos, seus inter-relacionamentos e o fluxo de mensagens entre eles;
* diagrama de atividades: representa a execução de ações ou atividades e os fluxos que são disparados pela conclusão de outras ações ou atividades;
* diagrama de máquina de estados: representa as ações ocorridas em resposta ao recebimento de eventos.

A UML e seus diagramas estão sendo utilizados nas etapas de desenvolvimento do *software* proposto. Tal desenvolvimento é apresentado na seção 6 deste artigo.

4. RUP (*Rational Unified Process*)

Segundo Kruchten (2003), o RUP é um processo de engenharia de *software* que fornece uma abordagem disciplinada para assumir tarefas e responsabilidades dentro de uma organização de desenvolvimento. Este processo tem com o objetivo assegurar a produção de *softwares* de alta qualidade que satisfaçam às necessidades de seus usuários finais dentro de prazo e orçamento previsíveis.

O RUP cobre o ciclo inteiro do processo de desenvolvimento do *software* e utiliza técnicas modernas e abordagens como tecnologia de objeto e desenvolvimento baseado em componente, modelagem e UML, arquitetura e desenvolvimento iterativo.

Esta metodologia fornece orientação sobre a ordem das atividades que uma equipe precisa desenvolver, quais artefatos devem ser desenvolvidos e quando isso deve se dar, orientando as tarefas individuais dos desenvolvedores e a equipe como um todo. O RUP também oferece critérios para monitoramento e medição dos produtos e atividades do projeto.

Abordando algumas práticas para desenvolvimento de *softwares*, o RUP visa dar qualidade ao projeto de construção do *software*, o contínuo monitoramento das atividades e um melhor desenvolvimento dos processos. As práticas são: desenvolver *softwares* iterativamente, gerenciar requisitos, usar arquiteturas baseadas em componentes, modelar visualmente o *software*, verificar continuamente a qualidade do *software* e controlar as mudanças do ambiente.

A metodologia RUP utiliza a abordagem iterativa para o desenvolvimento de *softwares*, na qual inicialmente se endereçam alguns requisitos e alguns riscos. Em seqüência parte do projeto e da implementação é construída e validada. Este processo é repetido até que se chegue ao final do projeto.

As iterações seguem um padrão semelhante à abordagem em cascata, onde o seu fluxo contém as atividades: extração de requisitos, análise, projeto, implementação, integração e teste. No entanto, o diferencial entre estas abordagens é a ênfase dada às atividades de uma iteração para a iteração da fase seguinte.

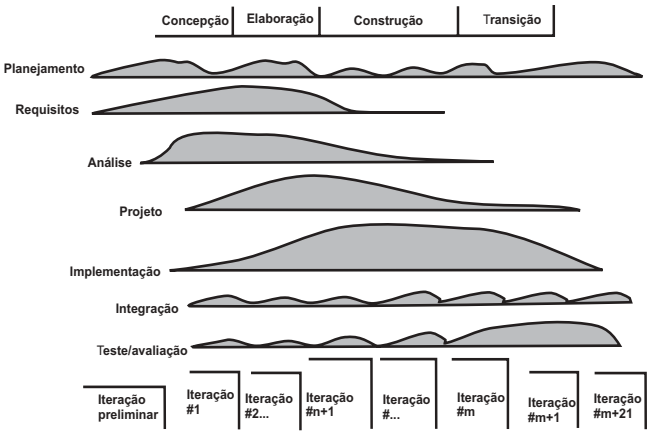

Figura 1. Importância das fases nas iterações. (KRUCHTEN, 2003)

A organização e a divisão das sucessões de iterações são baseadas nos objetivos específicos e no prazo. Os processos serão medidos pelo número de casos de uso completados, características completadas, casos de testes passados, requisitos de desempenho e riscos eliminados.

As etapas são divididas em fases e marcos. As fases do RUP e seus respectivos marcos são: concepção, objetivo do ciclo de vida, elaboração, arquitetura do ciclo de vida, construção, capacidade operacional inicial e lançamento de produto.

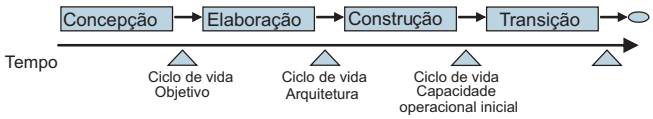

Figura 2 – Ciclo de Vida e Marcos do RUP. (KRUCHTEN, 2003)

* Concepção (fase): especifica o produto final, seu domínio de negócio e a extensão do projeto. Será concluída pelo marco objetivo do ciclo de vida (LCO).

* Objetivo do ciclo de vida (marco):
  - consentimento de interessados na definição de extensão e custo e estimativas de prazo.
  - compreensão dos requisitos como comprovado pela fidelidade dos casos de uso primários.
  - credibilidade do custo e estimativas de prazo; prioridades, riscos e processos de desenvolvimento.
  - profundidade e amplitude de qualquer protótipo arquitetônico que tenha sido desenvolvido
  - despesas atuais contra despesas planejadas.

* Elaboração (fase):
  - analisar o domínio do problema;
  - estabelecer uma fundamentação arquitetônica sadia;
  - desenvolver o plano de projeto;
  - eliminar os elementos de alto risco do projeto;
  - ser concluída pelo marco arquitetura do ciclo de vida (LCA).

* Arquitetura do ciclo de vida (marco):
  - A visão do produto é estável?
  - A arquitetura é estável?
  - A demonstração executável mostra os elementos de risco principais que foram endereçados e solucionados com confiança?
  - O plano da fase de construção é suficientemente detalhado e preciso? Ele é suportado por uma base confiável para gerar estimativas?
  - Todos os interessados concordam que a visão atual pode ser alcançada para desenvolver o sistema completo, se o plano corrente for executado no contexto da arquitetura atual?
  - A despesa de recurso atual *versus* a despesa planejada é aceitável?

* Construção (fase):
  - construir o produto;
  - evoluir a visão, a arquitetura e os planos até o produto;
  - Entregar à comunidade de usuários versões alfa e beta para teste.
  - Será concluída pelo marco a capacidade operacional inicial (IOC).

* Capacidade operacional inicial (marco):
  - O produto lançado é estável e está amadurecido o bastante para ser distribuído na comunidade de usuários?
  - Todos os interessados estão preparados para a transição à comunidade de usuários?
  - As despesas de recursos atuais *versus* despesas planejadas continuam aceitáveis?

*Transição (fase):
  - teste beta para validar o sistema novo contra expectativas do usuário;
  - operação paralela com o sistema de legado que o projeto está substituindo;
  - conversão de banco de dados operacionais;
  - treinamento de usuários e mantenedores;
  - saída do produto para *marketing*, distribuição e equipes de venda.
  - Será concluída pelo marco o lançamento do produto.

* Lançamento de produto (marco):
  - O usuário está satisfeito?
  - As despesas de recursos atuais as despesas de recursos planejadas ainda estão aceitáveis?

5. CARACTERÍSTICAS DO *SOFTWARE* EM DESENVOLVIMENTO

Um dos grandes problemas que muitas empresas enfrentam atualmente é o gerenciamento de sua comunicação interna. Distribuir de forma eficiente informações entre os grupos de trabalho de uma empresa vem se tornando uma tarefa complexa, principalmente em organizações com um número elevado de funcionários e de grupos de trabalho. O agendamento de atividades para grupos de trabalho tem se tornando uma tarefa difícil, uma vez que encontrar datas e horários de disponibilidade comum para estes grupos é uma tarefa nada fácil. Dentro deste contexto, o Grupo de Pesquisa em Sistemas de Informação do Cesumar (GPSI) decidiu desenvolver uma ferramenta para controlar de maneira automatizada e eficiente a comunicação interna de empresas (*Software* de Comunicação Corporativa).

O *software* a ser desenvolvido deverá suportar a manutenção e o gerenciamento conjunto dos seguintes tipos de agenda: geral (para toda a corporação), em grupo (específica) e individual (particular). A ferramenta oferecerá recursos para o agendamento de compromissos (individuais) e eventos (gerais), tais como: datas importantes, aniversários, prazo para entrega de relatórios/tarefas, reuniões, bem como compromissos e eventos personalizados às necessidades dos usuários.

Para oferecer suporte às novas tecnologias disponibilizadas atualmente (navegadores da internet e dispositivos móveis) o *software* deverá desenvolvido de forma a suportar independência de plataforma. Essa característica se refletiu de forma decisiva na decisão da linguagem de programação a ser utilizada.

Para realizar o desenvolvimento do *software* proposto foram tomadas algumas decisões de projeto, tais como:

- escolha do paradigma de desenvolvimento orientado a objetos, uma vez que este se apresenta, atualmente, como a metodologia de modelagem mais atrativa para *softwares* complexos; amplexos. Ae desenvolvimento Os principais vantagens dessa abordagem foram discutidas na seção 3 deste artigo;
- escolha do processo de desenvolvimento de *softwares* RUP, uma vez que, como destacado anteriormente, este modelo vem sendo amplamente utilizado em projetos de desenvolvimento de *softwares* orientados a objetos. Também se deve destacar que o modelo RUP prevê a construção de protótipos para o refinamento

de requisitos, característica esta desejada em virtude da falta de um conhecimento profundo do domínio do problema pelo GPSI;

- escolha da linguagem de programação Java para a implementação do *software*. Esta decisão justifica-se, pois Java é uma linguagem de programação orientada a objetos e foi projetada para diferentes plataformas e sistemas operacionais.

6. RESULTADOS OBTIDOS

No primeiro momento foram levantados os requisitos do sistema por meio de reuniões, em que foram realizadas a análise de negócios e a descrição dos processos. Nesta etapa todos os membros do GPSI tiveram oportunidade de fazer suas colocações e colaborar efetivamente para este levantamento.

Após os levantamentos de requisitos iniciais, foi desenvolvida a modelagem utilizando a notação e semântica do diagrama de caso de uso proposto na UML. As figuras 4 e 5 representam os diagramas elaborados e utilizados para entendimento e refinamento dos requisitos:

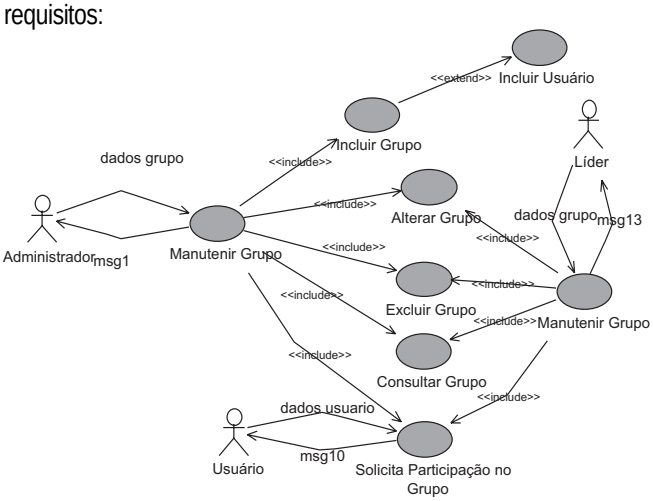

Figura 3 – Caso de uso Incluir Grupo

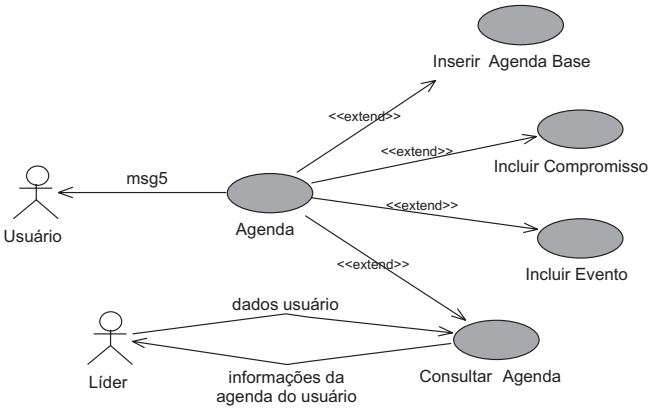

Figura 4 – Caso de uso Agenda

Após a diagramação dos casos de uso e a definição dos requisitos, foi preciso entender como esses requisitos ficariam disponíveis para o usuário. Logo se fez necessário demonstrar como supostamente seria o sistema, e isso também visou ao refinamento dos requisitos.

O RUP sugere que seja feito um ou vários protótipos na fase de concepção. Neste contexto, destaca-se a necessidade de prototipar para esclarecer e refinar os requisitos do sistema, para uma análise mais completa do escopo do sistema já na fase inicial de planejamento.

Segundo Kruchten (2003), os protótipos devem ser usados de forma direcionada a reduzir riscos e incertezas quanto à viabilidade do negócio de um produto que é desenvolvido e quanto à estabilidade ou desempenho da tecnologia-chave e ao compromisso ou disponibilidade do projeto.

Sommerville (2003) salienta que é difícil para os usuários finais prever como vão utilizar novos sistemas de *software* para dar apoio ao seu trabalho diário. Se esses sistemas forem grandes e complexos, provavelmente será impossível fazer essa avaliação antes de o sistema ser construído e colocado em operação. Como alternativa, é possível decidir deliberadamente construir um protótipo descartável para ajudar na análise e na validação de requisitos, sendo em seguida descartado o protótipo e construído um sistema com qualidade de produção.

De acordo com Pressman (2002), o cliente muitas vezes define um conjunto de objetivos gerais para o *software*, mas não identifica detalhadamente requisitos de entrada, processamento ou saída. Em outros casos, o desenvolvedor pode estar inseguro da eficiência de um algoritmo, da adaptabilidade de um sistema operacional ou da forma que a interação homem-máquina deve assumir. Nessas e em muitas outras situações, um paradigma de prototipagem pode oferecer a melhor abordagem.

Marçula e Benini Filho (2005) corroboram essa idéia dizendo que o protótipo deve ser criado com o propósito de cumprir os requisitos iniciais do usuário (ainda sem uma completa certeza da necessidade). O protótipo deve ser avaliado, refinado e construído novamente até que se obtenha a satisfação e se captem as necessidades do usuário.

Os protótipos construídos neste trabalho não tiveram como finalidade representar como o sistema seria em determinadas plataformas, uma vez que este deverá ser independente de plataforma, mas sim demonstrar os requisitos do sistema em forma de interface e melhorá-los através da avaliação do usuário. Dessa forma, as figuras 6 e 7 apresentam alguns dos protótipos já desenvolvidos para avaliação. Deve-se ressaltar que todos as principais funcionalidades foram representadas por meio de protótipos.

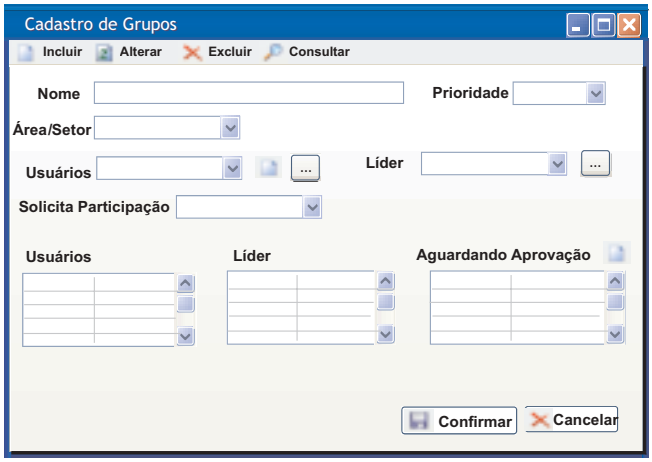

Figura 5 – Protótipo do processo cadastrar grupos

Figura 6 – Protótipo do processo *cadastrar agenda-base*

Uma vez que os requisitos iniciais foram descritos e modelados por meio de diagramas de caso de uso e subseqüentemente demonstrados e argumentados com a construção de protótipos, os casos de uso foram refinados e descritos.

O modelo utilizado para descrição dos casos de uso foi o sugerido por Medeiros (2004). Dentro deste contexto, as figuras 8 e 9 apresentam respectivamente a descrição de casos de uso dos processos *cadastrar grupo* e *cadastrar agenda-base*. É importante destacar que todos os casos de uso definidos para o *software* foram descritos.

Nome-referência
Cadastrar grupo.
Breve descritivo
Este caso de uso cria e mantém um grupo de usuários do sistema.
Pré-condições
Para cadastrar um setor, e necessário ter cadastrado no sistema pelo menos um setor, uma prioridade e dois usuários.
Atores envolvidos
Administrador, usuário e líder
Cenário principal
1. Incluir um grupo: o administrador informa o nome do grupo, sua prioridade, área/setor, o líder do grupo, os usuários que o compõem e confirmar os dados .2. Alterar um grupo: o líder ou o administrador escolhe o grupo que deseja alterar, e fornece os novos dados do grupo. 3. Excluir um grupo: o líder ou o administrador escolhe o grupo que deseja excluir e confirma a exclusão. 4. Solicita participação no grupo: o usuário, o líder ou o administrador o grupo que deseja participar e confirma a solicitação de participação.
Cenário alternativo
1.1 O usuário a qualquer momento da operação cancela a inclusão. 1.2 O usuário deixa de informar um dos dados exigidos e confirma (OK); será então exibida para o usuário uma mensagem informando qual informação precisa ser preenchida. 2.1 O usuário deixa de informar um dos dados exigidos e confirma (OK); será exibida para o usuário uma mensagem informando qual informação precisa ser preenchida. 3.1 O usuário tenta excluir um grupo que está em uso, e será mostrada uma mensagem informando que o grupo não pode ser excluído.
Requisitos Especiais
1.Verificar se já não existe um grupo semelhante para este setor.

Figura 7: Descrição do caso de uso cadastrar grupo

Nome-referência
Agenda-base.
Breve descritivo
Este caso de uso informa ao sistema a disponibilidade do funcionário para a empresa e suas atividades fixas anuais.
Pré-condições
O usuário deve informar suas atividades fixas anuais com a data e o período da ocorrência, informar os dias da semana em que trabalha e o período de trabalho em cada dia.
Atores envolvidos
Usuário
Cenário principal
1. Inserir e alterar a agenda-base: o usuário informa os dias da semana e o período em que ele trabalha para a empresa

em cada dia da semana e ainda informa suas atividades fixas na empresa. 2. O usuário confirma seus dados.
Cenário alternativo
1.1 O usuário a qualquer momento da operação cancela a inclusão dos dados, e sua agenda-base fica incompleta .2.1O usuário informa um valor inválido para um dos registros, o sistema mostra uma mensagem de erro informando qual registro não foi informado ou qual foi informado incorretamente.

Figura 8: Descrição do caso de uso agenda base

CONCLUSÕES

Este trabalho apresentou uma discussão sobre o uso do paradigma de orientação a objetos e do modelo de processo RUP para o desenvolvimento de um *software* corporativo.

A abordagem orientada a objetos foi utilizada por possibilitar um desenvolvimento estruturado, organizado e que permite um alto índice de reutilização de informações, e também por se apresentar como excelente solução para a modelagem de sistemas complexos, uma vez que esta metodologia possibilita a modularização de partes de um *software*.

Também foi discutida neste trabalho a importância de se terem requisitos bem-definidos antes de seguir para etapas mais avançadas do desenvolvimento. Para isso, utilizou-se a prototipação na fase de concepção do RUP. Este modelo de processo de *software* não impõe a construção de protótipos na fase de levantamento de requisitos, no entanto, pode-se observar que, quando não existe um grande conhecimento do domínio do problema, o uso deste artifício é extremamente útil.

Como próximos passos desta pesquisa espera-se concluir a etapa de elaboração que já foi iniciada e finalizar a implementação do *software* na fase de construção. As demais fases do modelo RUP exigirão a escolha de uma empresa para realização de um estudo de caso que descreva a implantação e avaliação do *software* desenvolvido neste trabalho.

REFERÊNCIAS

BECK, K.. **Embracing change with extreme programming.** IEEE Computer, n.10, v.32, p.70-78.

BOOCH, G., et. al. **UML, guia do usuário.** Rio de Janeiro: Campus, 2000.

D'SOUZA, D. F. et al. **Objects, Components and Frameworks with UML –** The Catalysis Approach. USA: Addison Wesley Publishing Company, 1999.

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

MARÇULA, M.; BENINI FILHO, P. A. **Informática**: conceitos e aplicações. São Paulo: Érica, 2005.

MEDEIROS, Ernani. **Desenvolvendo software com UML 2.0:** definitivo**.** São Paulo: Pearson Makron Books, 2004.

MELO, A. C.. **Desenvolvendo aplicações com UML 2.0:** do conceitual à implementação. 2. ed. Rio de Janeiro: Basport, 2004.

PRESSMAN, R. S. **Engenharia de Software**. 5. ed. Rio de Janeiro: McGraw-Hill, 2002.

SILVA, D. M. **Guia de consulta rápida:** UML**.** São Paulo: Novatec Editora, 2001.

SOMMERVILLE, I. **Engenharia de software.** 6. ed. São Paulo: Addison Wesley, 2003.