UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CURSO DE SISTEMAS DE INFORMAÇÃO

PROPOSTA PARA O GERENCIAMENTO DE CUSTOMIZAÇÕES DE SOFTWARE EM SEU PROCESSO DE DESENVOLVIMENTO

ALEXSANDRA DUARTE BORGES

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Sistemas de Informação.

Orientador: Hegler Correa Tissot
Co-orientador: Profº Leandro J. Komosinski

Florianópolis - SC
2008/2

Alexsandra Duarte Borges

PROPOSTA PARA O GERENCIAMENTO DE CUSTOMIZAÇÕES DE SOFTWARE EM SEU PROCESSO DE DESENVOLVIMENTO

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Sistemas de Informação.

Florianópolis, 05 de Novembro de 2008.

Orientador: Hegler Correa Tissot
Co-orientador: Profº Leandro J. Komosinski

Banca Examinadora:

Profª Patrícia Vilain

Profº Ricardo Pereira e Silva

Agradeço a Deus, por ter me permitido mais esta conquista.

À minha mãe querida, pela confiança, carinho e dedicação.

Ao meu orientador, Hegler C. Tissot, pela presença e orientação constantes.

Aos professores da UFSC que participaram dessa caminhada, em especial aos professores Leandro J. Komosinski, meu co-orientador, e Ricardo Pereira e Silva, membro da banca, pelo incentivo e auxílio.

E a todos aqueles que direta ou indiretamente colaboraram tornando possível a elaboração deste trabalho.

RESUMO

O desenvolvimento de softwares de forma rápida e eficaz torna-se uma necessidade cada vez mais presente nas empresas desenvolvedoras de softwares. O mercado exige destas empresas a qualidade do produto final a um menor custo de produção. Uma abordagem que permita o uso de softwares já prontos, onde seja necessária apenas a sua adaptação possibilita o reuso e flexibilidade no atendimento às necessidades do cliente. Esse é o princípio da customização de softwares.

Este trabalho apresenta uma proposta para o gerenciamento de customizações de software realizadas durante todo o processo de desenvolvimento do software. A proposta segue um modelo de customização de softwares em níveis, que permite a criação de uma estrutura onde softwares são desenvolvidos a partir de outros projetos de softwares já prontos.

Baseado nessa proposta, o trabalho apresenta a aplicação do metamodelo de gerenciamento de customizações de software em partes de um projeto de software customizado a partir de alguns softwares prontos, utilizados como base a customização.

Palavras-chave: customização de software, reuso de software, engenharia de software, metodologias de desenvolvimento de software

ABSTRACT

The fast and efficient software development has been more and more present in companies of software development. Market requires these companies final product quality with low production costs. An approach that enables the use of ready software, when only its adaptation is necessary, allowing the reuse and flexibility on customer needs. This is the software customization principle.

This study presents a proposal for software customizations management during all software development process. The proposal follows a software customization model by levels, which allows creating a structure where the software is developed from other ready software projects.

Based on this proposal, the work presents the metamodel application of software customizations management as part of a custom software project from some other ready software, used as a customization base.

Key-words: software customization, software reuse, software engineering, methodology for software development

SUMÁRIO

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

ABREVIATURAS E SIGLAS

- CRM: *Customer Relationship Management*
- CEP: Código de Endereçamento Postal
- RUP: *Rational Unified Process*
- UML: *Unified Modeling Language*
- XP: *Extreme Programming*

1 INTRODUÇÃO

1.1 APRESENTAÇÃO

A rapidez com que circulam as informações, a disponibilidade de ferramentas e o surgimento contínuo de novas tecnologias permitiram o estabelecimento de um ambiente altamente competitivo para o mercado de software em que variáveis como: bom atendimento, custo, tempo e flexibilidade sejam o diferencial para que as empresas consigam conquistar novos clientes e manter os antigos fidelizados.

Segundo Roselino (2006, p. 29) há um grande esforço por parte dos engenheiros de software na procura de modelos maduros de desenvolvimento que permitam a racionalização em busca de ganhos de produtividade no processo de produção.

As empresas que desenvolvem software cada vez mais têm que realizar seu trabalho, o desenvolvimento do software, com rapidez e eficiência para não perder o cliente. Mesmo quando a empresa fornecedora do software já possua a solução (pacote de aplicativos) a ser adquirido pelo cliente, o que pode tornar o processo de implantação naturalmente mais rápido, cada um destes clientes possui características e necessidades específicas. Neste caso, ou a empresa limita a personalização das funcionalidades do produto existente, reduzindo o custo de mão-de-obra e conseqüentemente seu valor de venda ou a empresa eleva o valor do produto e gasta um tempo considerável para modificar a ferramenta e adequá-la as necessidades do cliente.

Tendo em vista o alto custo de operacionalização deste processo, a criação de um modelo que permita um adequado gerenciamento de customizações de software trará à equipe de desenvolvimento flexibilidade na geração de novas funcionalidades e na remodelagem de funcionalidades já existentes. Com isso, o ganho de produção, maleabilidade e tempo beneficiaria ambas as partes, tanto do fornecedor do produto, quanto do cliente que o adquire.

1.2 OBJETIVOS

1.2.1 Objetivo geral

O objetivo geral deste trabalho é apresentar uma proposta para o gerenciamento de customizações de projetos de software através de um metamodelo que possa ser aplicado dentro do contexto e das fases de um processo de desenvolvimento de software.

1.2.2 Objetivos específicos

Os objetivos específicos são:

- Analisar a customização de software no contexto de um processo de desenvolvimento;
- Propor um metamodelo para o gerenciamento das customizações de software;
- Apresentar a aplicação da proposta de gerenciamento de customizações de software com base em alguns projetos de software exemplo.

1.3 METODOLOGIA DE PESQUISA

Este trabalho está baseado na apresentação de um metamodelo para o gerenciamento de customizações de software e fundamentado nas etapas descritas a seguir:

A primeira etapa consistiu em avaliar através da literatura uma metodologia de desenvolvimento de software e as técnicas de engenharia de software utilizadas para o desenvolvimento de sistemas customizáveis.

Na segunda etapa foram analisadas as relações existentes entre os projetos de software com customização e a proposta de um modelo de customização em níveis, onde são utilizados projetos de software como base para geração de projetos customizados.

Com isso foi possível a criação de uma proposta de um metamodelo que pudesse ser utilizado no processo de desenvolvimento de um software para efetuar o gerenciamento das customizações do software.

Esse metamodelo foi aplicado em um exemplo de um software customizado, que serviu para testar e validar a utilização da proposta.

A Figura 1 exemplifica a metodologia utilizada.

Figura 1 Metodologia de pesquisa utilizada

1.4 ORGANIZAÇÃO DO TEXTO

Este trabalho está organizado em cinco capítulos. No Capítulo 2 são revisados os principais conceitos relacionados com ao desenvolvimento de software e a customização de software. O Capítulo 3 apresenta um modelo de customização de software, a relação entre projetos customizados e a proposta de gerenciamento de customizações. O Capítulo 4 apresenta um exemplo de aplicação dessa proposta; e o Capítulo 5, as conclusões, trabalhos futuros e considerações finais.

2 DESENVOLVIMENTO DE SISTEMAS DE INFORMAÇÃO

O desenvolvimento de em software, especialmente os que possuem um grande número de requisitos e terão um grande número de usuários, pode ser uma tarefa árdua e complexa. O uso de uma metodologia de desenvolvimento de software que permita o mínimo de organização e controle poderá facilitar esse processo. Neste capítulo será apresentada uma visão geral de sistemas de informação, uma metodologia de desenvolvimento de software, o RUP (*Rational Unified Process*), os conceitos relacionados à customização e a produção de software, bem como algumas medidas que apóiam o desenvolvimento customizado de software.

2.1 SISTEMAS DE INFORMAÇÃO

2.1.1 Conceitos

Segundo Campos (1994), um sistema de informação é uma combinação estruturada de informação, recursos humanos, tecnologia de informação e práticas de trabalho, ordenados de forma a assegurar a execução dos objetivos de uma organização.

Araújo (1995) afirma que sistemas de informação são aqueles que, de uma forma geral, simplificam a realização de processos de comunicação.

Laudon e Laudon (1996) definem um sistema de informação como uma combinação de componentes inter-relacionados que reúnem, armazenam e efetuam o processamento e a distribuição de informações para suportar o controle e a tomada de decisão nas organizações.

Segundo Rezende (1999) o principal objetivo dos sistemas de informação é auxiliar o processo de tomada de decisões na organização e geralmente eles apresentam as seguintes características:

- Grande volume de dados e informações;
- Complexidade de processamento;
- Muitos clientes e/ou usuários envolvidos;
- Contexto abrangente, mutável e dinâmico;

- Interligação de diversas técnicas e tecnologias;
- Emanados de planejamento estratégico e interligado à gestão da empresa;
- Suporte e auxílio aos processos de tomada de decisões empresariais;
- Focado nos negócios da empresa.

2.2 METODOLOGIAS PARA O DESENVOLVIMENTO DE SOFTWARE

Segundo Rezende (1999), uma metodologia completa comporta uma abordagem organizada para atingir um objetivo, através de passos pré-determinados. É um roteiro para desenvolvimento organizado de projetos. A metodologia deve ajudar no processo de desenvolvimento de projetos, de modo que os mesmos atendam, de maneira satisfatória, às necessidades do cliente ou usuário, com os recursos disponíveis e dentro de um prazo ideal definido em conjunto com os envolvidos.

Existem várias metodologias de desenvolvimento de software definidas na literatura de Engenharia de Software como *Rational Unified Process* (RUP), *Extreme Programming* (XP), entre outras. Elas comumente são divididas em duas categorias: as metodologias “tradicionais” e as metodologias “ágeis”.

Neste trabalho, são utilizadas as orientações da metodologia de desenvolvimento RUP no que se refere à definição de etapas de desenvolvimento de um software e artefatos a serem produzidos nestas etapas. Por isso, a metodologia RUP é abordada com mais detalhes neste trabalho. Entretanto, a proposta apresentada poderá ser adequada para aplicação em quaisquer das metodologias de desenvolvimento de software que incorporem o ciclo de vida clássico da engenharia de software que, segundo Pressman (1995) inicia no nível de sistema e avança ao longo da análise, projeto, codificação, teste e manutenção.

2.3 O RUP

2.3.1 Conceito

Segundo Kruchten (2003) o RUP é um processo de engenharia de software. Ele fornece subsídios de forma estruturada e disciplinada dentro de uma organização de desenvolvimento de softwares, para que sejam designadas tarefas e responsabilidades. Seu objetivo é assegurar da melhor forma possível, a produção de software de qualidade e que atenda as necessidades de seus usuários finais dentro de um prazo e custo previsíveis.

A arquitetura geral do RUP, representada na Figura 2, é composta por um processo com duas dimensões:

- O eixo vertical representa os aspectos ou fluxos essenciais do processo, que agrupa as atividades por tipo. Essa dimensão representa o aspecto estático do processo, suas atividades, fluxos e artefatos.
- O eixo horizontal representa o tempo e mostra as diversas fases em que se desdobra o processo. Essa dimensão mostra o aspecto dinâmico, quais suas fases, ciclos e iterações.

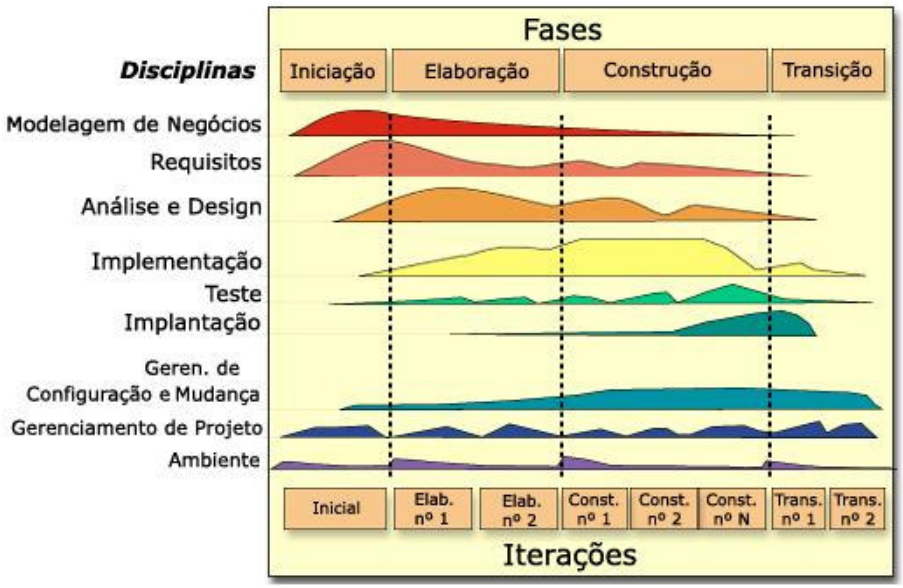

Figura 2 Estrutura do processo RUP
Fonte: RATIONAL, 2002

Segundo Kruchten (2003), o RUP integra muito das melhores práticas de desenvolvimento de software, mais especificamente seis delas são cobertas: desenvolver software iterativamente, gerenciar requisitos, usar arquiteturas baseadas em componentes, modelar visualmente, verificação contínua da qualidade e controle de mudanças.

2.3.2 As Fases

O processo iterativo é organizado em ciclos que por sua vez é composto de fases, representadas na Figura 3. São quatro fases que interagem entre si durante o tempo de duração do ciclo: Concepção, Elaboração, Construção e Transição.

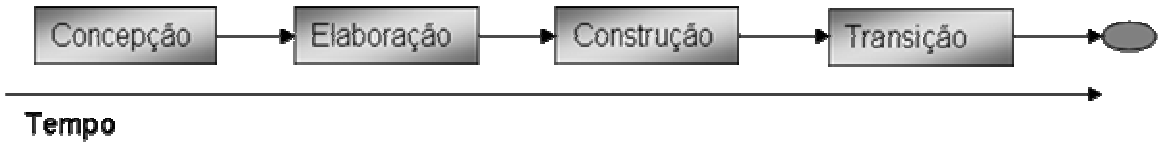

Figura 3 Fases de Desenvolvimento
Fonte: KRUCHTEN, 2003

2.3.2.1 Concepção

Nesta fase é definido o escopo (domínio e extensão) do projeto, as limitações, os riscos e o custo global do projeto. O importante nesta etapa é a concordância de todos os envolvidos acerca das definições iniciais do projeto.

As atividades essenciais nesta fase são a elaboração do documento que contém a visão geral, os principais requisitos, características e restrições, um glossário inicial do projeto, os possíveis casos de uso, uma avaliação de riscos e um plano de projeto com todas as fases.

2.3.2.2 Elaboração

O objetivo desta fase é a análise do domínio do problema, planejamento das atividades e recursos necessários, bem como a especificação das características e projeto da arquitetura do sistema e a eliminação dos elementos de alto risco.

Nesta etapa são gerados o modelo de casos de uso, uma descrição da arquitetura de software, um plano de desenvolvimento global e descrição dos requisitos adicionais.

2.3.2.3 Construção

Durante a fase de construção os componentes e características restantes são integrados ao produto final. O que marca esta fase é o gerenciamento de recursos, controle e otimização de processo, desenvolvimento e teste dos componentes de software.

Os resultados esperados desta fase são o produto de software, testado e integrado às plataformas requeridas, os manuais de usuário e o lançamento inicial do produto.

2.3.2.4 Transição

Nesta fase concluí-se que o software ou parte dele já está em nível que qualidade aceitável para ser submetido à comunidade de usuários. Os treinamentos dos usuários são realizados e o software passa pela sua avaliação, existindo a possibilidade de correções, alterações ou melhorias do produto, dando início a um novo ciclo de desenvolvimento.

2.3.3 Fluxos de Processo e Artefatos de Software[1]

Artefatos de software são produtos de trabalho produzidos e utilizados durante o desenvolvimento dos projetos. Eles são utilizados para registrar e capturar informações do projeto de software. Segundo Rational (2002) um artefato de software pode ser:

- Um **documento**, como no caso de um Documento de Arquitetura de Software ou um Caso de Negócio
- Um **modelo**, como no caso de um Modelo de Casos de Uso

[1] O termo *artefato de software* é usado neste trabalho de forma genérica, referindo-se a qualquer produto produzido durante o processo de desenvolvimento do software.

- Um **elemento do modelo** como um Caso de Uso de um Modelo de Casos de Uso.

Segundo Kruchten (2003), no RUP os artefatos de software são organizados em grupos que correspondem aos Fluxos de Processo, que são nove:

- Fluxo de gerenciamento de projeto
- Fluxo de modelagem de negócio
- Fluxo de requisitos
- Fluxo de análise e construção
- Fluxo de implementação
- Fluxo de teste
- Fluxo de gerenciamento de configuração e mudanças
- Fluxo de ambiente
- Fluxo de distribuição

Alguns artefatos de software são produzidos em um fluxo, mas também são utilizados em outros fluxos durante o processo de desenvolvimento. Os próximos itens descreverão sucintamente alguns destes fluxos de processo e alguns dos artefatos de software produzidos no fluxo.

2.3.3.1 O fluxo de modelagem de negócio

O fluxo de modelagem de negócio tem por meta principal compreender a organização para qual o sistema será desenvolvido, bem como suas necessidades, e garantir que essa compreensão seja compartilhada por todos os envolvidos, ou seja, clientes, usuários finais e desenvolvedores.

Para alcançar as metas elencadas, o fluxo de modelagem de negócio descreve como desenvolver uma visão da nova organização alvo e, baseando nessa visão, definir os processos, papéis e responsabilidades daquela organização em um modelo de negócio (KRUCHTEN, 2003). Neste caso, o uso de técnicas de modelagem nessa etapa facilita a compreensão e funciona como um meio de relacionar o domínio do negócio com o domínio do software, uma vez que eles estão

usando a mesma linguagem. A Figura 4 mostra os papéis[2] envolvidos e os artefatos produzidos no fluxo de modelagem de negócio.

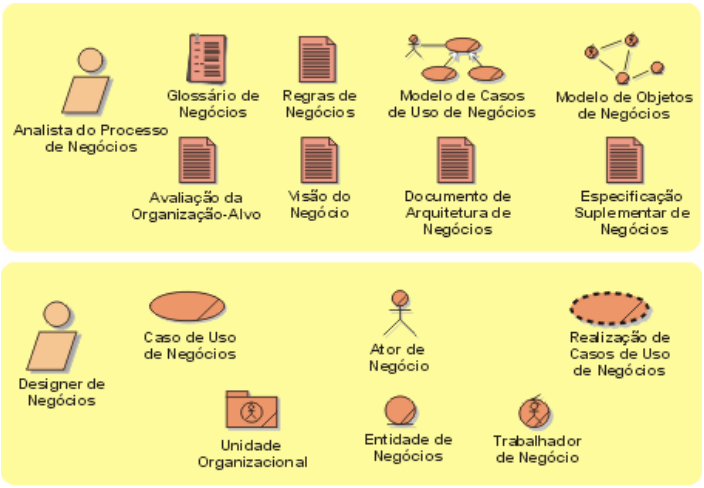

Figura 4 Papéis envolvidos e Artefatos de Software produzidos no Fluxo de Modelagem de Negócio
Fonte: RATIONAL, 2002

Visão do Negócio

A Visão do Negócio é um documento que visa descrever a estrutura e a dinâmica da organização e dos envolvidos. Um documento de Visão de Negócios possui informações como: descrições dos envolvidos e dos clientes, perfis dos envolvidos, perfis dos clientes e objetivos da modelagem de negócios.

Glossário de Negócios

O Glossário de Negócios é um documento que contém os mais importantes termos utilizados na modelagem de negócios do projeto, bem como a sua definição.

2.3.3.2 O fluxo de requisitos

Pressman (1995) diz que uma compreensão total e completa dos requisitos de software é imprescindível para um desenvolvimento de software bem-sucedido. Ou seja, não há méritos em desenvolver um software excelente, mas que não atenda as

[2] Os papéis são definições abstratas de um conjunto de atividades executadas por uma pessoa ou grupo de pessoas (RATIONAL, 2002). Durante as atividades que os artefatos de software são produzidos.

reais necessidades do cliente. Entender estas necessidades é essencial para que o resultado final seja satisfatório para ambas as partes.

O fluxo de requisitos descreve como definir os requisitos de software detalhadamente através de uma visão do sistema representada por um modelo de casos de uso e especificações adicionais. A Figura 5 mostra os papéis envolvidos e os artefatos produzidos no fluxo de requisitos.

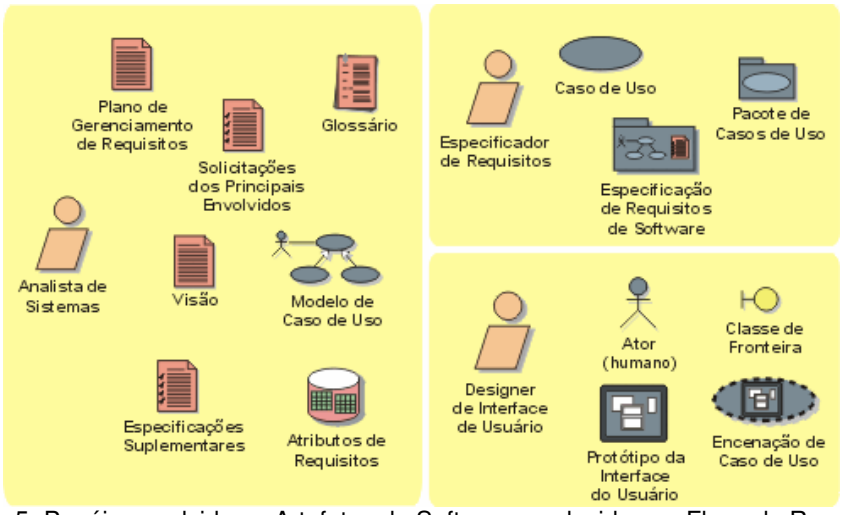

Figura 5 Papéis envolvidos e Artefatos de Software produzidos no Fluxo de Requisitos
Fonte: RATIONAL, 2001

Modelo de Casos de Uso

O Modelo de Casos de Uso é um modelo das funções pretendidas do software. No RUP ele é muito importante, principalmente como fonte de informações para a análise, projeto e teste, e exerce a função de contrato estabelecido entre o cliente e os desenvolvedores (RATIONAL, 2002). A Figura 6 mostra um exemplo de diagrama de casos de uso.

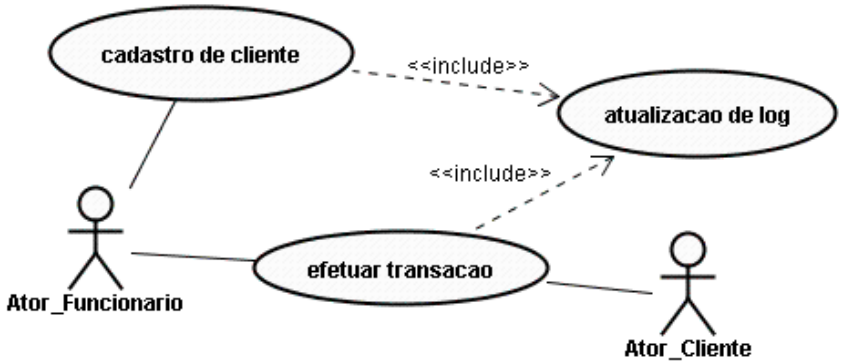

Figura 6 Representação de um Diagrama de Casos de Uso
Fonte: SILVA, 2007

Plano de Gerenciamento de Requisitos

O Plano de Gerenciamento de Requisitos é um documento que contém os tipos de requisitos, a definição, descrição e atributos dos requisitos do sistema. Ele também contém a descrição dos mecanismos que serão utilizados para o controle e rastreabilidade das alterações dos requisitos ao longo do projeto.

2.3.3.3 O fluxo de análise e projeto

O objetivo principal do fluxo de análise e projeto é transformar e condensar os requisitos em uma especificação que permita a implementação do sistema (KRUCHTEN, 2003). Durante este processo será definido o modelo de projeto, a arquitetura do software, bem como os componentes desta arquitetura.

O projeto de software deve definir o sistema de forma que este seja implementado sem erros de interpretação e de forma que atenda todos os requisitos contidos na especificação de requisitos. O grau de detalhamento do projeto pode ter variações de empresa para empresa ou de projeto para projeto.

A Figura 7 mostra os papéis envolvidos e os artefatos produzidos no fluxo de análise e projeto.

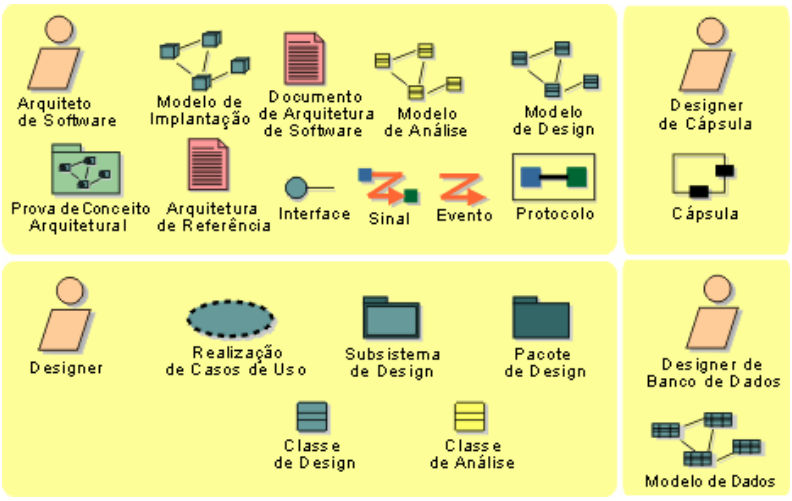

Figura 7 Papéis envolvidos e Artefatos de Software produzidos no Fluxo de Análise e Projeto
Fonte: RATIONAL, 2001

Documento de Arquitetura de Software

O documento de Arquitetura de Software deve fornecer uma visão geral da arquitetura do sistema. Ele pode ser composto por diferentes visões da arquitetura como: visão de casos de uso, visão lógica, visão de dados, entre outras.

Modelo de Dados

O modelo de dados fornece uma visão lógica ou física dos dados persistentes no sistema. Sua abrangência inclui a definição de qualquer comportamento relativo ao banco de dados como procedimentos, índices, visões, entre outros. A Figura 8 mostra um exemplo de um modelo de dados físico.

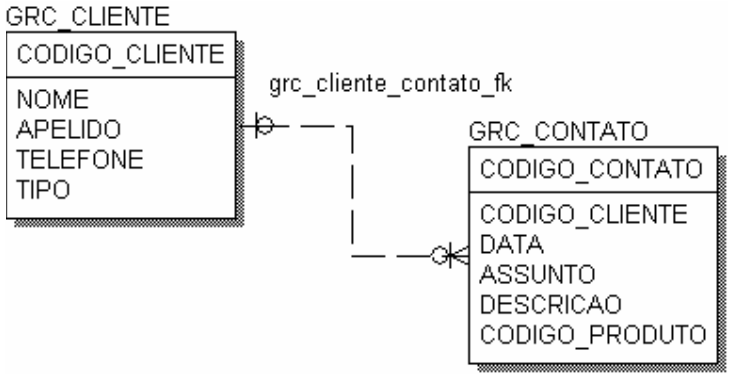

Figura 8 Modelo de Dados físico

2.3.3.4 O fluxo de implementação

É no fluxo de implementação que é definida a organização do código (as camadas, os subsistemas e os componentes), são produzidos os arquivos-fonte, binários, executáveis e outros, são efetuados os testes de unidade e a integração dos resultados produzidos por implementadores individuais ou equipes. O RUP trabalha no fluxo de implementação com três conceitos: Construções, Integração e Protótipos. A Figura 9 mostra os papéis envolvidos e os artefatos produzidos no fluxo de implementação.

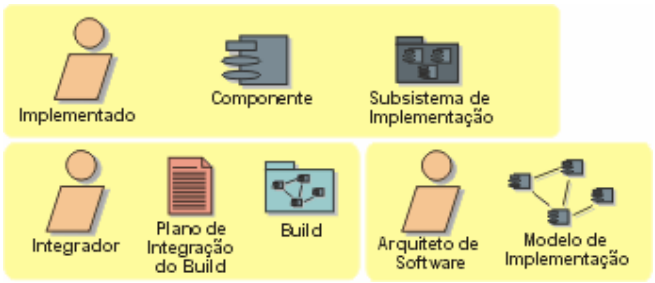

Figura 9 Papéis envolvidos e Artefatos de Software produzidos no Fluxo de Implementação
Fonte: RATIONAL, 2001

Componente

Um componente pode ser um trecho de código, um conjunto de outros componentes ou mesmo um arquivo de inicialização. Ele representa uma parte física do processo de desenvolvimento: arquivos de algum tipo que contêm código-fonte, configurações, executáveis e outros produtos físicos do desenvolvimento (RATIONAL, 2002).

2.3.3.5 O fluxo de teste

Segundo Kruchten (2003), a proposta do teste é avaliar a qualidade do produto, o que não só envolve o produto final como inicia logo no projeto com a avaliação da arquitetura e continua pela avaliação do produto final entregue aos clientes. O teste é executado sobre tipos diferentes de alvos em fases diferentes do desenvolvimento de software.

Os estágios que envolvem o fluxo teste são o teste de unidade, de integração, de sistema e de aceitação. O teste de unidade é o teste executado sobre elementos do

sistema passíveis de testes individuais. O teste de integração é o teste executado sobre unidades do sistema integradas. O teste de sistema é o teste executado sobre a aplicação completa. Já o teste de aceitação é o teste realizado pelos usuários, sobre toda a aplicação, com a finalidade de determinar se o sistema pode ser distribuído.

A Figura 10 mostra os papéis envolvidos e os artefatos produzidos no fluxo de teste.

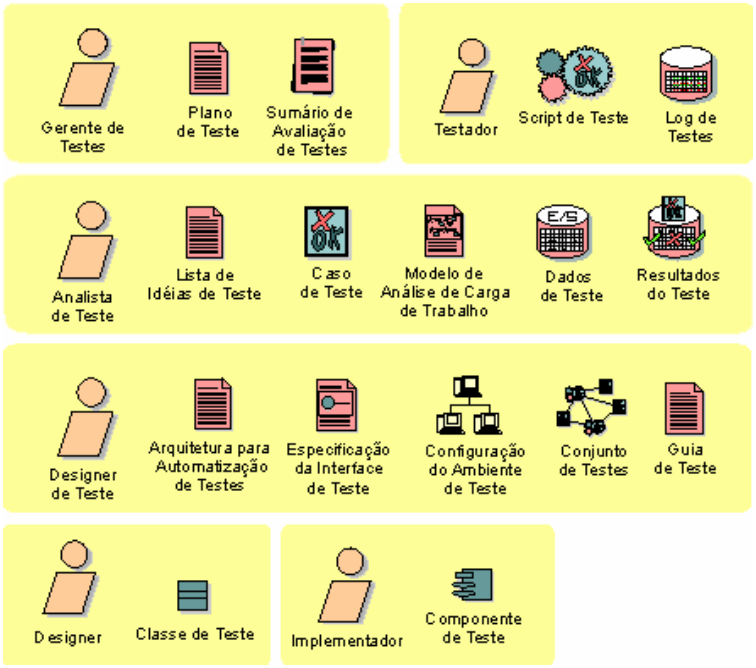

Figura 10Papéis envolvidos e Artefatos de Software produzidos no Fluxo de Teste
Fonte: RATIONAL, 2001

Plano de Teste

O plano de teste é um documento que contém as metas e objetivos dos testes, a abordagem, os tipos e técnicas de teste que serão utilizados, os itens de teste alvo, entre outros.

Classe de Teste

A classe de teste é um estereótipo de Classe do modelo de design. Sua finalidade é identificar a funcionalidade específica de teste para facilitar ou automatizar testes (RATIONAL, 2002).

2.4 CUSTOMIZAÇÃO DE SOFTWARE

> *Customização tem o sentido de adaptar os produtos e processos ao gosto do cliente, portanto é o atendimento que visa a satisfação do freguês. A origem da palavra está no inglês customer, que significa cliente.*
> Maria Tereza de Queiroz Piacentini, em 2008.

O termo customização passou a existir na língua portuguesa há pouco tempo. É um neologismo criado para exprimir personalização. Sua origem também pode ser associada ao termo em inglês "*custom*" que quer dizer feito sob medida ou personalizado. O termo, bem como aplicação do conceito vem sendo utilizado nos vários setores da indústria e de serviços, como a customização de roupas, de automóveis, etc.

Customizar um software é redesenhá-lo ou adequá-lo às necessidades de quem o irá usar. No desenvolvimento essa customização pode ser realizada de duas formas distintas: já no processo de concepção e análise (na definição do software) ou com o produto já pronto, quando apenas algumas características e funcionalidades são modificadas para que este se molde aos novos requisitos.

No mercado de produção e comercialização de softwares, existem diferentes formas de relacionamento entre o cliente e o desenvolvedor do software. Diferentes formas de se produzir e comercializar o software para um ou mais clientes:

- O desenvolvimento para um cliente específico, com requisitos próprios onde o cliente procura a empresa desenvolvedora de softwares, relata suas dificuldades e a contrata para resolver seu problema no que diz respeito à tecnologia da informação.
- O desenvolvimento para um perfil de cliente ou segmento específico de mercado onde não existe um cliente "físico" que acompanhe o processo e tenha requisitos próprios. Nesta abordagem a empresa desenvolvedora de softwares identifica uma necessidade ou carência do mercado e investe no desenvolvimento de uma ferramenta que atenda esse nicho de mercado.

A primeira abordagem, segundo Roselino (2006, p. 36) é classificada como "serviço em software" ou também "software serviço" e é destinada ao atendimento da demanda que não é satisfeita por softwares prontos existentes no mercado. Já a

segunda, onde já existe um produto de software, o relacionamento cliente/empresa pode acontecer de formas distintas:

- O software pode ser totalmente fechado para quaisquer alterações, o chamado “software de prateleira”;
- Ou o software pode servir como base para o desenvolvimento de um novo software. Este voltado a um cliente específico que demandará as necessidades não atendidas pelo software base. Ou seja, o software se adaptará ao cliente através da sua customização.

Esta última abordagem, onde existe um software base para a criação de um software customizado que atenda as necessidades de um cliente específico, é o princípio do modelo de customização que será adotado para a proposta de gerenciamento de customizações de software. A proposta de gerenciamento, bem como o modelo de customização é detalhada no capítulo 3.

2.5 MEDIDAS QUE APÓIAM O DESENVOLVIMENTO CUSTOMIZADO DE SOFTWARES

2.5.1 Manutenibilidade

Manutenibilidade, segundo Rezende (1999), é “a facilidade com que um programa pode ser corrigido, se um erro for encontrado; adaptado, se o seu ambiente se modificar ou ampliado, se o cliente desejar inclusões e alterações nos requisitos funcionais”.

Segundo Nomura (2001 *apud* COAD, 1993), um número grande de empresas e organizações reconhece que entre 75% e 80% do custo de um sistema ocorre depois de ele ter sido implantado e estiver em operação; o que significa que muitas correções são efetuadas e funcionalidades acrescidas depois da implantação do sistema. Desta forma, um método que enfatize a manutenção melhora a produtividade da empresa por permitir aos profissionais de manutenção efetuar modificações mais rapidamente.

Geralmente, na literatura especializada, a manutenção de software é classificada em três tipos: a corretiva, a adaptativa e a perfectiva. A corretiva seria para corrigir erros no software. A adaptativa são modificações feitas com o objetivo de acompanhar as modificações do ambiente em que o software deve operar. Já a perfectiva se relaciona à melhoria do produto em função da evolução das necessidades do usuário. E, como um software dificilmente pode ser considerado ideal, inevitavelmente ele precisará de um ou mais dos três tipos de manutenção citados. A Figura 11 mostra os percentuais de esforço aplicados em cada um dos três tipos de manutenção apresentados.

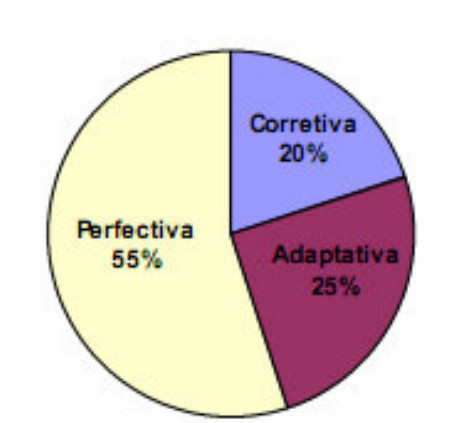

Figura 11 Percentuais de esforço de manutenção
Fonte: PIGOSKI, 1996

De acordo com Pressman (1995), “a maioria dos problemas associados à manutenção de software pode remeter-se a deficiências na maneira segundo a qual o software foi planejado e desenvolvido”. A ausência de um controle nas etapas do desenvolvimento do software vão se traduzir em problemas durante a manutenção do software.

A negligência no projeto, implementação e teste do software pode acarretar na dificuldade de manutenção deste, uma vez que correções ou adaptações em sua estrutura (manutenção corretiva, preventiva ou adaptativa) podem gerar um custo final muito superior ao esperado.

Dentre os fatores que podem impactar na manutenibilidade, segundo Brusamolin (2004), estão a arquitetura, a tecnologia e a documentação utilizadas no projeto. A arquitetura que, quanto melhor projetada nas fases iniciais do desenvolvimento, pode trazer um custo menor no momento da manutenção; a tecnologia, que tem que permitir um bom grau de manutenibilidade; e por fim a documentação e a

compreensibilidade do programa, fatores fundamentais para entendimento da estrutura e do funcionamento do software que sofrerá a manutenção.

Embora o termo manutenibilidade seja difícil de se quantificar, podemos avaliá-la indiretamente, através dos atributos da atividade de manutenção. Pressman (1995) apresenta uma série de métricas da manutenibilidade que se relaciona ao esforço despendido durante a manutenção propostas por Gilb (1979), que são:

- Tempo de reconhecimento do problema.
- Tempo de retardo administrativo.
- Tempo de coleta de ferramentas de manutenção.
- Tempo de análise do problema.
- Tempo de especificação das mudanças.
- Tempo de correção (ou modificação) ativa.
- Tempo de testes globais.
- Tempo de revisão de manutenção.
- Tempo de recuperação total.

Por fim, práticas de manutenibilidade poderiam ser pensadas e aplicadas já nas fases iniciais de desenvolvimento de um software. Uma vez que as necessidades dos usuários e o ambiente podem mudar constantemente e, com isso, o sistema sofrerá modificações, quanto maior for o grau de manutenibilidade deste, menor poderá será o esforço aplicado no processo de manutenção.

2.5.2 Reusabilidade

Segundo Rezende (1999), a reusabilidade é a medida que um programa (ou partes de um programa) pode ser reusado em outras aplicações – relacionada ao empacotamento e escopo das funções que o programa executa.

Para Silva (2000, p.13 *apud* JOHNSON, 1988 e PREE, 1994):

> A reutilização de software, em contraposição ao desenvolvimento de todas as partes de um sistema, é um fator que pode levar ao aumento da qualidade e da produtividade da atividade de desenvolvimento de software. Isto se baseia na perspectiva de que o reuso de artefatos de software já desenvolvidos e depurados, reduza o tempo de desenvolvimento, de testes e as possibilidades de introdução de erros na produção de novos artefatos. A reutilização de artefatos de software em larga escala é um dos

> argumentos a favor da abordagem de orientação a objetos. Em muitos casos, porém, constitui uma perspectiva frustrada, pois a reutilização não é característica inerente da orientação a objetos, mas é obtida a partir do uso de técnicas que produzam software reutilizável.

A reusabilidade é um dos fatores associados aos conceitos de utilização de frameworks e desenvolvimento orientado a componentes. A possibilidade de se reutilizar artefatos de software no desenvolvimento pode reduzir consideravelmente o custo geral do projeto, tanto em tempo quanto em esforço na codificação. Este ganho de tempo poderá beneficiar outras fases do projeto, como a de testes ou mesmo implantação.

2.6 CONCLUSÃO

Neste capítulo foram apresentados os conceitos relacionados à produção de software, metodologias de desenvolvimento de software, em especial o RUP. Foram também apresentadas as etapas de desenvolvimento de um software segundo a metodologia de desenvolvimento RUP e alguns artefatos produzidos em cada uma das etapas, conceitos relacionados à customização de software, bem como algumas medidas que apóiam o desenvolvimento customizado.

3 PROPOSTA PARA GERENCIAMENTO DA CUSTOMIZAÇÃO DE SOFTWARE NAS ETAPAS DO DESENVOLVIMENTO

Artefatos de software são produzidos em todas as etapas de desenvolvimento de um software, seja na fase de análise, de testes ou mesmo da implantação. Em um projeto de software são produzidos diversos artefatos, e uma das tarefas que fazem parte do projeto é gerenciamento desses artefatos durante todo o processo de desenvolvimento.

Quando se tem uma situação em que um software deve ser alterado para gerar uma nova versão desse mesmo software, adaptando algumas de suas funcionalidades, pode ser necessário um trabalho extra de gerenciamento dos artefatos de software e das dependências que venham a ser geradas entre os artefatos de um projeto em relação ao projeto base.

Gerenciar artefatos de software, seguindo um modelo de customização de projetos em diversos níveis pode se tornar uma tarefa árdua se não existir um processo que permita o gerenciamento das interdependências entre os artefatos de um projeto customizado em relação ao seu projeto base.

Neste capítulo será descrita a ligação de um projeto de software e seus elementos, um modelo de customização de software e uma proposta para o gerenciamento em todas as etapas de desenvolvimento do software das customizações realizadas.

3.1 PROJETOS DE SOFTWARE: ARTEFATOS E ELEMENTOS

Em um projeto de software, como abordado no capítulo 2, diversos tipos de artefatos de software são produzidos. E um artefato de software, segundo a abordagem deste trabalho, é todo e qualquer elemento que compõe o projeto, que é produzido durante o processo de desenvolvimento do software. Ou seja, um Diagrama de Classes é um artefato de software, assim como uma única classe deste diagrama. Desta forma, na proposta apresentada (capítulo 3) e na aplicação da proposta (capítulo 4) todos os artefatos de software produzidos para um projeto são tratados genericamente como elementos do projeto. No entanto, cada tipo de

elemento deve ter uma definição específica de acordo com a metodologia de desenvolvimento utilizada[3].

A relação existente entre um projeto e seus elementos está ilustrada na Figura 12, através de um diagrama de classes.

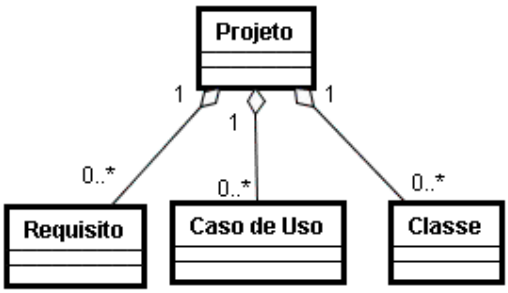

Figura 12 Diagrama de Classes de um Projeto de Software

A Figura 13 apresenta um diagrama de objetos para exemplificar o diagrama de classes da Figura 12.

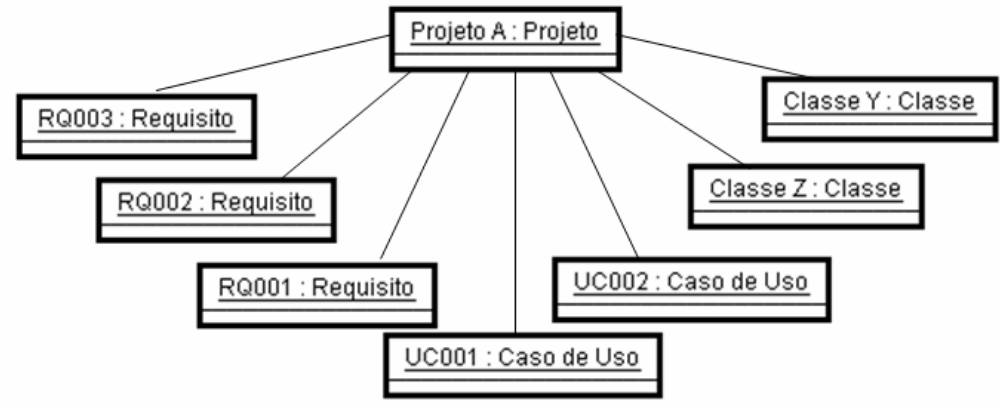

Figura 13 Diagrama de Objetos de um Projeto de Software

A metodologia utilizada no processo de desenvolvimento orienta na escolha dos tipos de elementos a serem gerados para cada etapa do projeto, bem como os atributos ou propriedades necessários para cada um desses tipos de elementos.

A Figura 14 ilustra um exemplo onde:

- Um requisito X é um objeto da classe Requisito, associado ao projeto Y, objeto da classe Projeto.

[3] Neste trabalho são utilizadas as orientações do RUP para a escolha dos tipos de elementos que compõem o projeto de software, assim como a UML para representação dos diagramas e modelos.

- A classe Requisito possui os atributos identificação, definição e descrição.
- A classe Projeto possui os atributos nome e data de início.

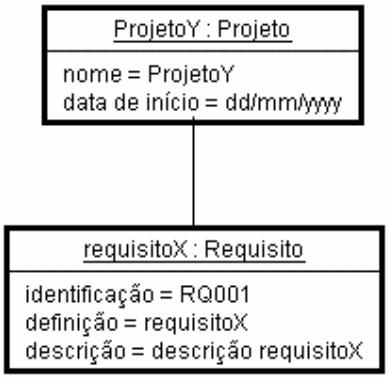

Figura 14 Diagrama de Objetos representando os objetos ProjetoY e requisitoX
Fonte: do autor

O requisitoX é um elemento (objeto) associado ao ProjetoY. Assim como o requisitoX, outros elementos, inclusive de tipos diferentes, podem estar associados ao ProjetoY.

3.2 UM MODELO DE CUSTOMIZAÇÃO DE SOFTWARE

O objetivo de se fazer customizações de software, segundo a abordagem deste trabalho, vem do princípio de se utilizar um ou mais softwares como base para construção/desenvolvimento de novos projetos de software. Ou seja, utilizar-se de softwares base onde o usuário final apontaria quais suas necessidades dentro do que já existe (o que pode ser usado) e quais requisitos teriam que ser adicionados e/ou modificados. Essa abordagem estaria em utilizar parte do projeto de software pronto e incrementá-lo ou modificá-lo às solicitações do cliente como mostra a Figura 15. Neste caso, cada versão customizada para diferentes clientes possui todas as características do software base e mais as características e requisitos que não são contemplados por este, e que são criados e modelados especificamente em cada versão.

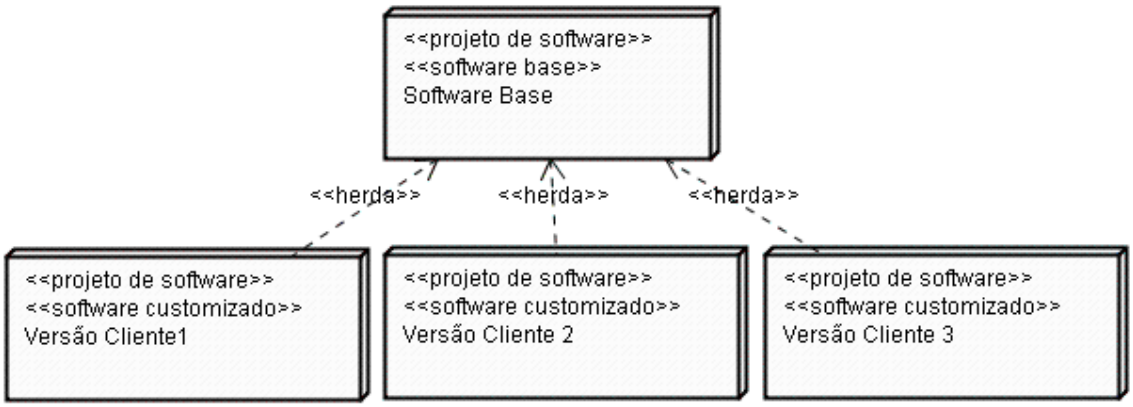

Figura 15 Diagrama de Utilização representando a Customização de Projetos de Software

Existem casos em que apenas o uso de um software base não satisfaz as necessidades do cliente. É quando mais de um software base pode estar presente na composição de um único projeto.

Quando existe uma necessidade de um sistema que efetue mais de um tipo de controle ou atenda a características diversas, o software que será desenvolvido, o produto final, é uma combinação de pequenos sistemas que atendem separadamente os diversos tipos de controle a serem efetuados. Pode ser usado, neste caso, mais de um software base para construção/desenvolvimento do software customizado, como representado na Figura 16.

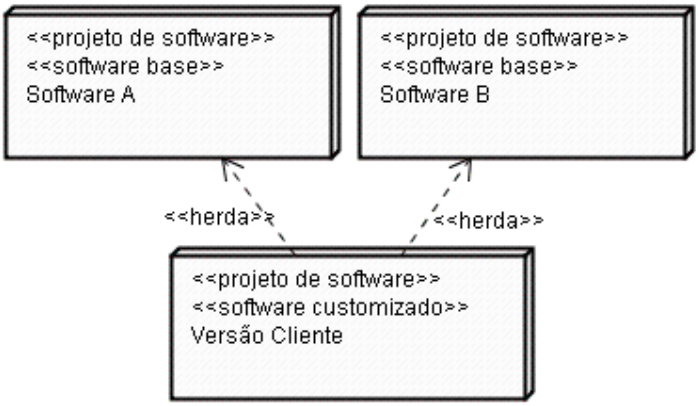

Figura 16 Software customizado que utiliza dois softwares base

Embora existam projetos base que sirvam apenas a nichos específicos de mercado, também existe aqueles que servem muito bem a áreas diversas, como relacionamento com o cliente ou correspondência eletrônica, ou mesmo projetos

menos complexos como um sistema de autenticação de usuário ou de *log*. Na estrutura representada na Figura 16, por exemplo, poderia haver um segundo cliente com o mesmo tipo de negócio que desejasse apenas um tipo de controle, ou seja, o Software base A não seria utilizado, como mostra a Figura 17.

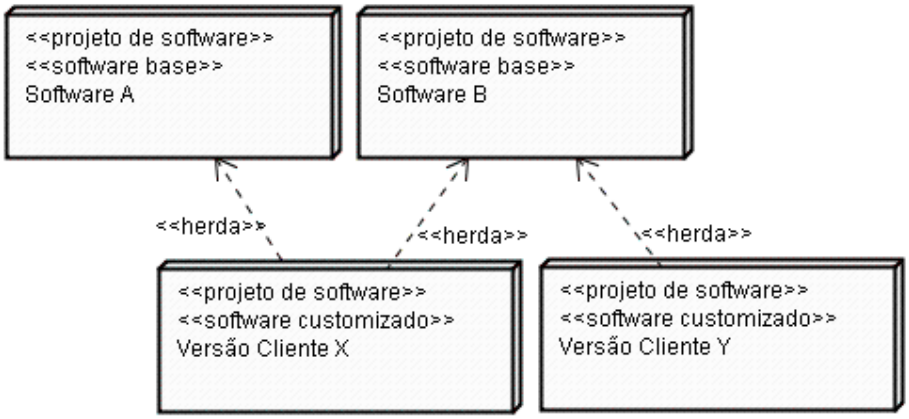

Figura 17 Softwares customizados para dois clientes distintos que utilizam dois e um softwares base

Projetos concebidos para suportar customização, segundo o modelo descrito, formam uma relação de dependência e herança com seus projetos base. Quando há uma modificação no “Software Base B” (Figura 17), os projetos dos clientes X e Y, que usam este software base, também poderão receber estas modificações (caso isto esteja previsto no projeto). A forma como isso acontecerá, o gerenciamento da customização do software será abordado mais adiante neste capítulo. O importante no momento é compreender como se dá a customização e a relação de dependência e herança existente entre os projetos customizados e customizáveis.

Partindo do princípio de que cada software customizado é criado a partir de projetos de software base, e que estes tem uma relação de dependência com os projetos utilizados para compô-lo, podemos chegar a um nível em que os próprios projetos customizados podem servir de base para outros projetos. Ou seja, um projeto de um cliente X pode servir como base de um projeto de um cliente Y e assim sucessivamente. Um projeto de software, neste caso, poderá em um determinado momento ser software customizado (gerado por um ou mais de um software base) e em outro momento ser software base para outro projeto. A Figura 18 representa uma estrutura em que projetos de software base compõem projetos customizados que se transformam em software base para novos projetos.

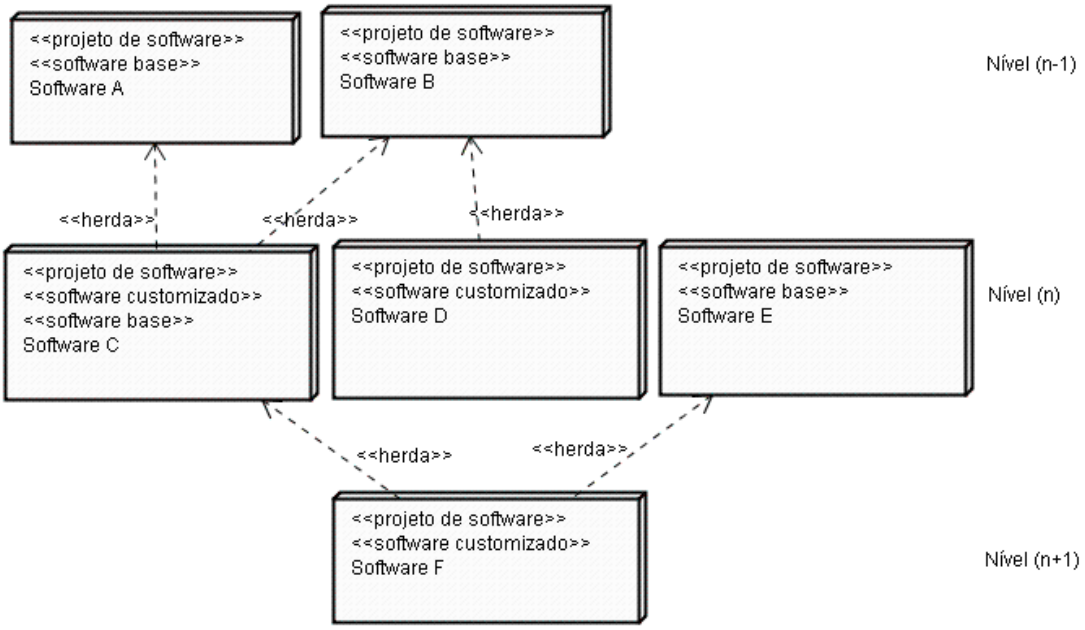

Figura 18 Diagrama de Utilização de customizações de projetos de software em níveis I

A Figura 18 demonstra a hierarquia existente entre os projetos com customização. Embora alguns projetos de software sirvam apenas como base (os que estão no topo da hierarquia), outros projetos de software serão base ou customizados dependendo do nível a ser considerado. Se for considerado o nível em que se encontra o “Software C”, nível (n), os projetos de software que estão acima, nível (n-1), são projetos base para o “Software C”; já os que estão abaixo, nível (n+1), são projetos customizados a partir do “Software C”. Como existem projetos de software em um nível acima do nível (n), os projetos de software que estão neste nível, nível (n), são também considerados customizados, identificados genericamente como P(n). Já os projetos de software que estão no nível acima (n-1), que são neste caso projetos base e/ou customizáveis são identificados genericamente como P(n-1).

Já no caso de ser considerado o nível em que se encontra o “Software F” como P(n), o nível em que está o “Software C” e o “Software E” será o P(n-1) e o nível em que está o “Software A” o nível P(n-2), conforme Figura 19. Neste caso, os projetos “Software C” e “Software E” são customizáveis e base para o “Software F”, que é customizado.

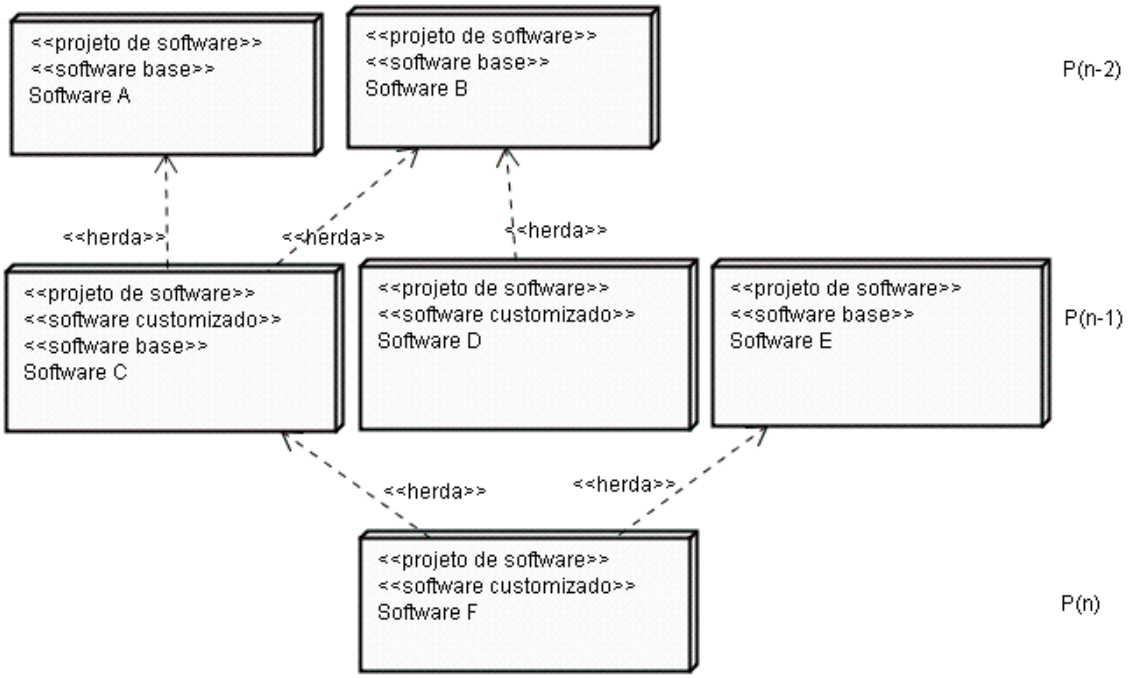

Figura 19 Diagrama de Utilização de customizações de projetos de software em níveis II

A customização de software, conforme o modelo de customização descrito pode gerar uma árvore com diferentes níveis de hierarquia, em que um projeto de software P(n) herda características de outro projeto P(n-1). Ou seja, todos os artefatos e elementos produzidos no projeto P(n-1) são herdados pelo projeto P(n). É criado, a partir deste momento, um vínculo de dependência entre os projetos. Ao se realizar alterações nos elementos do projeto P(n-1) os elementos do projeto P(n) herdados devem incorporar essas alterações. O vínculo existente entre projetos em diferentes níveis na hierarquia de customização é representado na classe Projeto, do diagrama de classes da Figura 12, por um auto-relacionamento nessa classe, conforme Figura 20.

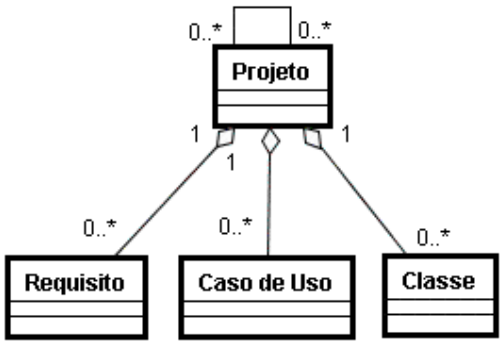

Figura 20 Diagrama de Classes de Projetos com Customização I

A Figura 20 mostra que um projeto pode ser dependente ou dependido por mais de um projeto. Isso quer dizer que um projeto P(n) pode ser desenvolvido com base em um ou mais projetos customizáveis P(n-1), ao mesmo tempo em que pode servir de base para a customização de novos projetos P(n+1).

Quando um projeto P(n-1) é usado como base para o projeto customizado P(n), este herda todos os elementos do projeto P(n-1), criando também uma relação de dependência entre os elementos de ambos os projetos. O elemento do projeto customizado P(n), que é um elemento herdado, é dependente do elemento do projeto base P(n-1). E, assim como nos projetos, cada classe de elemento terá um auto-relacionamento indicando possíveis vínculos com elementos de outros projetos. A Figura 21 apresenta o diagrama de classes de projetos de software com customização. Já a Figura 22 apresenta um diagrama de objetos que mostra como seria a associação dos objetos de projetos e elementos em 2 níveis da hierarquia de customização.

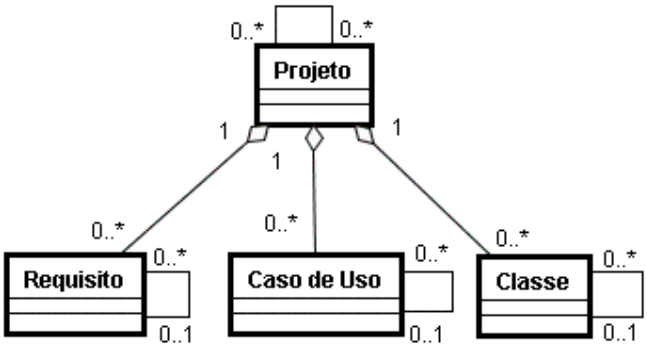

Figura 21 Diagrama de Classes Projetos com Customização II

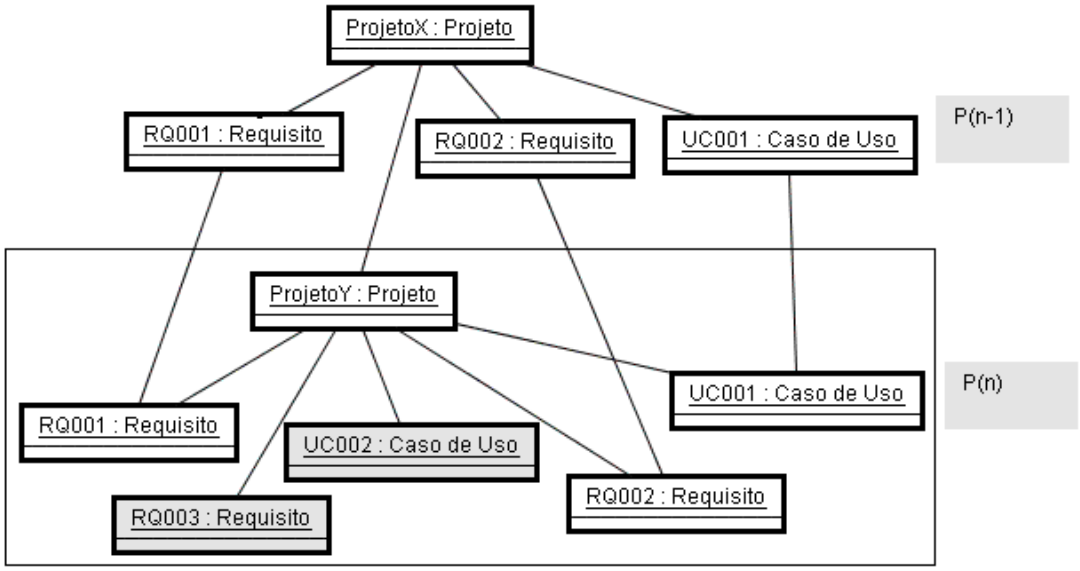

Figura 22 Diagrama de Objetos de Projetos com Customização

Na Figura 22 o ProjetoX está no nível (n-1) e é um software base para o ProjetoY, que está no nível (n). Cada um dos elementos do ProjetoX foi herdado pelo ProjetoY, exceto os elementos criados apenas para o ProjetoY, que neste caso estão apenas no nível (n): os objetos RQ003 e UC002 das classes Requisito e “Caso de Uso”, respectivamente.

Os diagramas apresentados na Figura 20 e na Figura 21 apresentam como seria a estrutura de projetos modelados para serem customizáveis, mas não contém ainda elementos que permitam gerenciar adequadamente as customizações realizadas nestes projetos.

3.3 GERENCIAMENTO DA CUSTOMIZAÇÃO

De uma forma prática, ao se customizar um software, na verdade, estão sendo realizadas customizações nas propriedades dos elementos que compõem esse software, ou seja, nos atributos das classes dos diferentes tipos de elementos.

A idéia central da proposta apresentada neste capítulo é incorporar o gerenciamento das customizações no próprio modelo que gerencia os elementos produzidos, permitindo, assim, que a customização de um elemento possa, ao

mesmo tempo, ser identificada e rastreada nos diversos projetos onde esse elemento pode estar presente customizado ou não.

Um requisito – que é um elemento de projeto de software – é herdado por outro projeto de software e neste outro projeto este requisito é modificado, ou seja, customizado. Para gerenciar a customização realizada e saber de onde o requisito foi herdado é preciso identificar na especificação do requisito a qual projeto ele pertence e se suas propriedades originais foram alteradas. E, para isso, é necessário que os elementos de software incorporem propriedades que identifiquem sua origem e situação no projeto. Ou seja, além das propriedades inerentes a cada elemento, definidas pelo desenvolvedor ou metodologia utilizada, o elemento deve possuir ainda propriedades que permitam realizar o gerenciamento da sua customização.

Propõe-se então, a geração de um modelo de classes, baseado no modelo de classes de projetos apresentado no item 3.1, para o gerenciamento das customizações realizadas no software. Este modelo poderá ser estendido por projetos em que se deseje realizar o gerenciamento de customizações de software. Esse gerenciamento poderá ser aplicado e mapeado nos três diferentes níveis que compõem um projeto de software:

- Projeto
- Elemento
- Propriedade

3.3.1 Projeto

O gerenciamento da customização de projetos trata da relação existente entre projetos base e projetos customizados. Isso será administrado por uma classe de gerenciamento, denominada CProjeto, que possui informações referentes à customização e à extensibilidade do projeto de software.

Esta classe deverá ser estendida pela classe Projeto para que seja possível efetuar o gerenciamento de customizações dos projetos e, desta forma, além dos atributos inerentes a um projeto de software, o projeto terá atributos referentes à sua customização.

Os atributos da classe CProjeto estão definidos na Tabela 1.

Tabela 1

Atributos da classe CProjeto

Atributo	Tipo de Valor
Identificação	Texto
Extensível	Sim/Não
Customizável	Sim/Não
Projetos base	Lista de Projetos
Projetos customizados	Lista de Projetos

Atributos da classe CProjeto

- **Identificação** – Este atributo é a identificação do projeto.
- **Extensível** – Este atributo indica se o projeto pode ser utilizado como base de outro projeto.
- **Customizável** – Este atributo indica se o projeto de software poderá sofrer modificações no projeto do software customizado. Caso o valor seja “Não”, os elementos, bem como o valor das propriedades dos elementos deste projeto não poderão ser modificados no projeto do software customizado.
- **Projetos Base** – Este atributo contém uma lista dos projetos-base utilizados pelo projeto customizado.
- **Projetos Customizados** – Este atributo contém uma lista dos projetos customizados a partir deste projeto, em que ele foi base.

A partir da definição dos atributos da classe CProjeto, da qual os projetos são estendidos, pode-se visualizar na Figura 23 como fica o diagrama de classes de um projeto com customização. O auto-relacionamento, antes existente na classe Projeto para indicar a herança entre os projetos com customização, foi substituído pelos atributos “Projetos base” e “Projetos customizados”, conforme Figura 23.

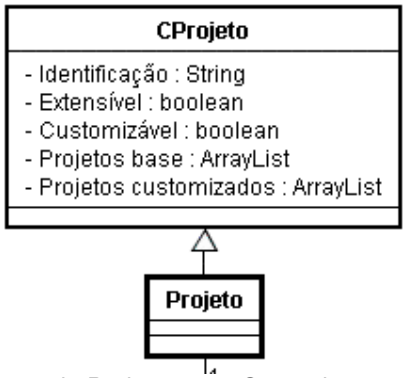

Figura 23 Diagrama de Classes de Projetos com Gerenciamento de Customização (Projeto)

A classe Projeto, como representado na Figura 23, é uma extensão da classe CProjeto. Sendo assim, todos os objetos da classe Projeto, além das propriedades ou atributos de um projeto, também terão as propriedades da classe CProjeto. A Figura 24 mostra um objeto da classe Projeto já contendo as propriedades para o gerenciamento da customização do software.

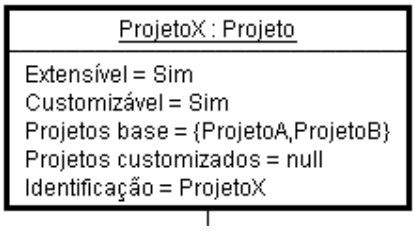

Figura 24Objeto da classe Projeto – extensão da classe CProjeto

3.3.2 Elemento

O gerenciamento de customizações do elemento de software trata a relação existente entre um elemento herdado e seu elemento base. Isso será administrado por uma classe de gerenciamento, denominada CElemento, que possui informações referentes a customizações do elemento de software.

Esta classe deverá ser estendida por todas as classes de elemento de software para que seja possível efetuar o gerenciamento de customizações dos elementos e, desta forma, além dos atributos inerentes a um tipo específico de elemento, o elemento terá atributos referentes à sua customização.

Os atributos da classe CElemento estão definidos na Tabela 2.

Tabela 2

Atributos da classe CElemento

Atributo	Tipo de Valor
Identificação	Texto
Customizável	Sim/Não
Elemento base	Objeto de CElemento
Elementos customizados	Lista de elementos (CElemento)

Atributos da classe CElemento

- **Identificação** – Este atributo é a identificação do elemento no projeto.
- **Customizável** – Este atributo indica se o elemento poderá sofrer modificações no projeto do software customizado. Caso o valor seja “Sim”, o valor das propriedades do elemento poderá ser modificado no projeto do software customizado que herdar este elemento.
- **Elemento base** – Este atributo é a referência para o elemento do qual o elemento do software customizado foi herdado, ou seja, para o elemento do projeto de software base.
- **Elementos customizados** – Este atributo contém uma lista dos elementos customizados a partir deste elemento, em que ele foi herdado.

A partir da definição dos atributos da classe CElemento, da qual os diferentes tipos de elementos são estendidos, podemos visualizar na Figura 25 como ficaria o diagrama de classes de um projeto com customização. Como a relação entre os elementos herdados (de um projeto customizado) e os elementos do projeto base deve ser gerenciada pela classe CElemento, o auto-relacionamento, antes existente na classe respectiva do elemento para indicar a herança entre os elementos, foi substituído pelos atributos “Elemento base” e “Elementos customizados” na classe CElemento. E a dependência de um elemento com customização para um projeto com customização é representada por uma dependência da classe CElemento com a classe CProjeto, conforme Figura 25.

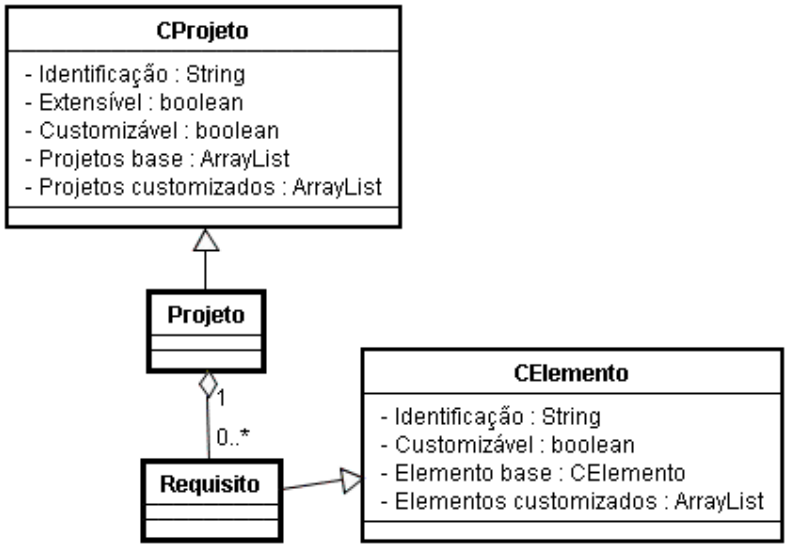

Figura 25 Diagrama de Classes de Projetos com Gerenciamento de Customização (Elemento)

A classe de elemento de software Requisito (Figura 25) é uma extensão da classe CElemento. Sendo assim, todos os objetos da classe Requisito, além dos atributos de um requisito, também terá os atributos da classe CElemento. A Figura 26 mostra um objeto da classe Requisito já contendo os atributos para o gerenciamento da customização do elemento.

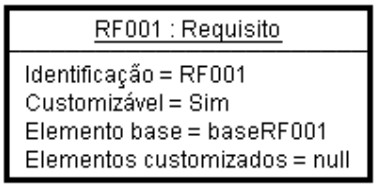

Figura 26 Objeto da classe Requisito – estendendo a classe CElemento

3.3.3 Propriedade

Embora os atributos da classe CElemento permitam identificar a situação do elemento no projeto, tanto no projeto herdado P(n-1) quanto no projeto que herdou P(n), as modificações ou customizações propriamente ditas são realizadas nos atributos próprios à cada tipo de elemento. E este tipo de nível de gerenciamento ainda não foi contemplado. É necessário identificar a customização de cada uma das propriedades e/ou atributos definidos para cada tipo de elemento de artefato de

software. No caso da classe Requisito, se o objeto RF001 representado na Figura 26 pertencesse à um projeto base P(n-1) e fosse herdado por outro P(n), o valor dos atributos definidos para o tipo de elemento Requisito poderia ser modificado no projeto P(n) se esse elemento fosse customizável.

O gerenciamento da customização deve indicar se algum atributo de algum elemento herdado foi alterado em relação ao projeto do qual foi herdado. Isso é possível se forem incorporadas informações de customização a cada propriedade de cada elemento do projeto. Para isso, é necessário abstrair as propriedades dos elementos de software e generalizá-las na figura de uma classe, a classe Propriedade, onde seus atributos são: identificação e valor. Cada tipo/classe de elemento do projeto estará associada à classe Propriedade e criará quantas referências forem necessárias para definição de suas propriedades. Esse novo modelo, baseado no modelo de classes apresentado na Figura 21, está ilustrado na Figura 27.

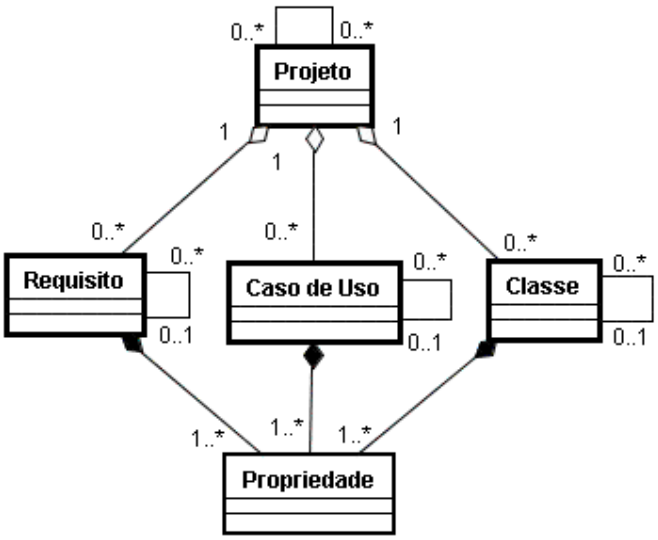

Figura 27 Diagrama de Classes de Projetos com Customização III

O gerenciamento das customizações das propriedades dos elementos trata a conformidade do valor da propriedade do elemento herdado em relação ao valor da propriedade correspondente no elemento base. Isso será administrado por uma classe de gerenciamento, denominada CPropriedade, que substituirá a classe Propriedade descrita anteriormente e terá as informações referentes à propriedade e suas customizações.

Os atributos da classe CPropriedade estão definidos na Tabela 3.

Tabela 3

Atributos da classe CPropriedade

Atributo	Tipo de Valor
Identificação	Texto
Valor	Texto
Valor Herdado	Sim/Não

Atributos da classe CPropriedade

- **Identificação** – Este atributo é a identificação da propriedade no elemento
- **Valor** – É o valor da propriedade do elemento.
- **Valor Herdado** – Este atributo indicará se o valor da propriedade é herdado (se é o mesmo valor da propriedade do elemento base) ou se não (quando a propriedade foi criada no projeto ou o valor da propriedade foi customizado).

Na Figura 27 a classe Propriedade está associada às classes de tipos de elementos. Ao ser substituída a classe Propriedade pela classe CPropriedade, a responsabilidade de gerenciar as propriedades de um elemento passou a ser da classe CPropriedade. Como a classe CPropriedade é responsável também pelo gerenciamento das customizações das propriedades, o relacionamento de composição antes existente entre a classe Propriedade e as classes de elementos é transferido para as classes CPropriedade/CElemento, conforme Figura 28. Uma vez que cada elemento é uma extensão da classe CElemento, cada tipo de elemento poderá instanciar quantas e quais propriedades forem necessárias ao seu tipo específico.

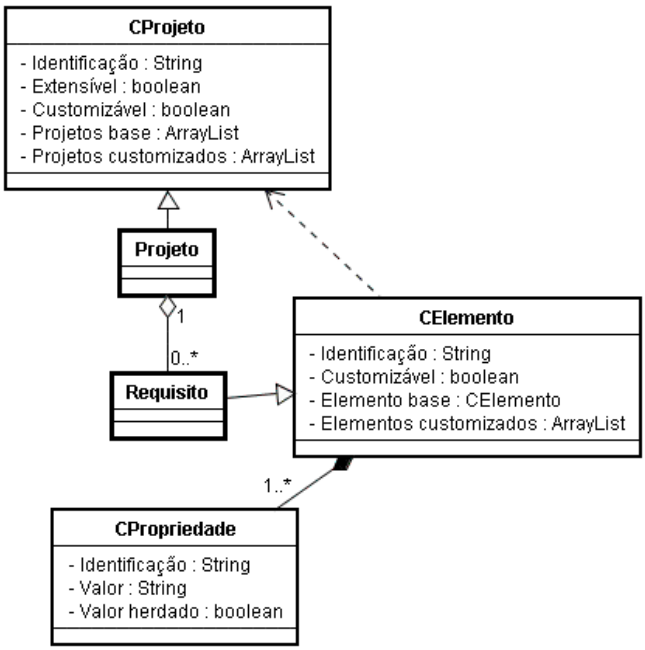

Figura 28 Diagrama de Classes de Projetos com Gerenciamento de Customização (Propriedade)

A classe CPropriedade (Figura 28) está associada à classe CElemento, da qual o tipo de elemento (classe) Requisito é estendido. Cada propriedade de elemento será do tipo CPropriedade. A Figura 29 mostra como ficará o diagrama de objetos das classes Projeto e Requisito com a adaptação de atributos (de exemplo) da classe Requisito para objetos da classe CPropriedade. Para isso, serão considerados dois atributos de exemplo para a classe Requisito: definição e descrição.

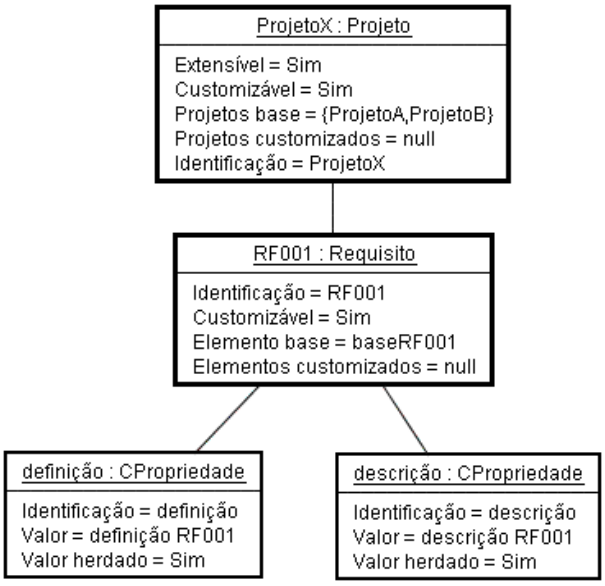

Figura 29 Diagrama de Objetos de um Projeto Customizável

3.3.4 Restrições para o Gerenciamento de Customizações de Softwares

Como são muitos os elementos de artefatos de software a serem gerenciados em um projeto de software e as propriedades, tanto as de customização quanto as inerentes a cada tipo de elemento podem ou não ser modificadas de um projeto para outro, é importante que se crie um procedimento para modificação dos valores dessas propriedades. Mesmo que um elemento tenha sido herdado de um projeto P(n-1), existem casos em que, por definição do projetista do software base, este elemento não pode ser customizado no projeto do software que o herda P(n). Neste caso, é necessário definir regras para a manutenção da integridade do modelo, bem como a propagação de modificações realizadas nos projetos dos softwares modelo P(n-1) aos projetos dos softwares customizados P(n).

O gerenciamento de customizações em projetos de software, por este motivo, deve atender às seguintes regras:

3.3.4.1 Em relação ao Projeto

Regra 1 Quando um projeto está definido como “não extensível” (atributo Extensível de CProjeto=Não) ele não poderá servir de base para outros projetos. Essa regra só será aplicada para novos projetos customizados. Os projetos já associados ao projeto definido como “não extensível” continuarão associados ao projeto;

Regra 2 Um projeto pode estar associado a mais de um projeto, seja no nível (n+1) ou (n-1), exceto pelo descrito na Regra 1;

Regra 3 Quando um projeto é definido como “não customizável” (atributo Customizável de CProjeto=Não), os seus elementos manterão o valor da propriedade Customizável, no entanto, também serão considerados “não customizáveis”;

Regra 4 Quando um projeto está definido como customizável (atributo Customizável de CProjeto=Sim), ao ser utilizado como base por outro projeto, o valor do atributo Customizável dos elementos (CElemento) herdados terá o mesmo valor do elemento do projeto base;

Regra 5 Quando um projeto é base para outro, todos os seus elementos e propriedades (valor da propriedade) são herdados, identificando no elemento de qual projeto ele foi herdado;

Regra 6 O vínculo entre um projeto base P(n-1) e um projeto customizado P(n) pode ser destruído. Neste caso o projeto customizado P(n) e os seus elementos passam a não depender mais do projeto base P(n-1) e de seus elementos;

Regra 7 Quando um projeto P(n-1) deixa de ser base de outro, todos os elementos do projeto customizado P(n) herdados do projeto base P(n-1) podem ser mantidos no projeto customizado, mas o vínculo com o elemento base é quebrado e o valor do atributo "Valor Herdado" das propriedades (CPropriedade) dos elementos tem seu valor alterado para "Não";

Regra 8 Quando um projeto tem o valor de seu atributo Customizável alterado, as características anteriores à alteração são mantidas. Ou seja, se o valor do atributo for alterado para "Não", os elementos do projeto passam a ser considerados também "não customizáveis" embora o valor do atributo Customizável de CElemento seja mantido. Neste caso, se o elemento for customizável (valor do atributo Customizável=Sim) deverá ser verificada a situação do projeto, se ele também é customizável.

3.3.4.2 Em relação ao Elemento

Regra 9 Cada elemento pertence a um projeto específico, e pode estar associado a apenas outro elemento de outro projeto (quando herdado);

Regra 10 Elementos de um projeto podem estar associados apenas a elementos de projetos associados ao seu projeto base P(n-1);

Regra 11 Apenas elementos indicados como "customizáveis" (atributo Customizável de CElemento=Sim) podem ter o atributo Customizável e os valores de suas propriedades alteradas em projetos herdados P(n), exceto quando o atributo de propriedade "Valor Herdado" estiver indicado como Não. Neste caso, o valor da propriedade poderá ser modificado independente do elemento base estar indicado como customizável;

Regra 12 O elemento, ao ser herdado de outro projeto, mantém todas as características de seu elemento base, exceto o atributo de propriedade (CPropriedade) "Valor Herdado" que inicialmente terá seu valor definido como Sim.

Regra 13 O vínculo entre um elemento de um projeto base P(n-1) e um elemento de um projeto customizado P(n) pode ser destruído. Neste caso, ou o elemento do projeto customizado P(n) é destruído ou é mantido e passa a não depender mais do elemento do projeto base P(n-1). O vínculo é quebrado;

Regra 14 Quando um elemento de um projeto customizado P(n-1) deixa de ser base de outro, o elemento do projeto customizado P(n) herdado do projeto base P(n-1) poderá ser mantido no projeto customizado, mas o vínculo com o elemento base é quebrado e o valor do atributo "Valor Herdado" das propriedades do elemento (CPropriedade) tem seu valor alterado para Não;

Regra 15 Quando um elemento de um projeto tem o valor de seu atributo Customizável alterado, as características anteriores à alteração são mantidas. Ou seja, se o valor do atributo for alterado para Não, as propriedades dos elementos herdados não poderão mais ser modificadas, exceto quando ela já foi customizada (o valor do atributo "Valor Herdado" de CPropriedade=Não);

Regra 16 Ao ser criado um novo elemento em um projeto. Se o projeto possuir projetos customizados (é base para outros projetos), o mesmo elemento será criado nos projetos customizados, com as mesmas características do elemento criado, exceto o atributo de propriedade "Valor Herdado" que inicialmente terá seu valor definido como Sim;

Regra 17 Um elemento de um projeto poderá ser destruído, inclusive elementos herdados. Ou seja, quando um elemento de um projeto base é destruído, os elementos customizados dos projetos P(n+n) podem:

a) ser destruídos da mesma forma que o elemento base, ou

b) serem mantidos, neste caso, quebrando o vínculo entre os elementos e mantendo nos elementos E(n+m) os valores das propriedades originais de E(n-1) que eram herdadas.

3.3.4.3 Em relação à Propriedade

Regra 18 Valores de propriedades que estão indicados como customizados (propriedade "Valor Herdado" de CPropriedade=Não) não considerarão o valor da propriedade do elemento do projeto base P(n-1) – elemento base;

Regra 19 Valores de propriedades que estão indicados como "não customizados" (propriedade "Valor Herdado" de CPropriedade=Sim) considerarão o valor da

propriedade do elemento do projeto base P(n-1) – elemento base, exceto pelo descrito na Regra 18;

Regra 20 Quando um projeto é herdado por outro (usado como base), todas as propriedades (valor das propriedades) são herdadas conseqüentemente. A propriedade "Valor Herdado" de CPropriedade, neste caso, inicialmente, será por padrão definida como Sim;

Regra 21 Ao ser alterado um valor de propriedade de um elemento herdado de outro projeto, o valor da propriedade "Valor Herdado" de CPropriedade deverá automaticamente ser alterada para Não.

Regra 22 Propriedades herdadas de outros elementos (projetos base) que tiveram seus valores customizados (alterados) poderão retornar ao valor original (valor da propriedade do elemento base). Neste caso, o valor do atributo de CPropriedade "Valor Herdado" será definido como Sim.

O modelo de gerenciamento de customizações de software apresentado, composto por classes de gerenciamento associadas ao projeto, aos elementos e às suas propriedades, permite rastrear as customizações realizadas durante todo o desenvolvimento de um software, bem como o relacionamento entre projetos de softwares customizados e customizáveis. Essa proposta, no entanto, não foi projetada para permitir a administração e o registro da evolução histórica das diferentes versões de cada elemento ou artefato.

Quando um software é implantando em um cliente e há, posteriormente, a necessidade da realização de manutenções deve existir um cuidado por parte da empresa desenvolvedora do software de manter os elementos do projeto devidamente atualizados. Ou seja, não deve haver modificações fora do gerenciamento de customizações, pois isso geraria uma versão do software diferente da versão gerenciada pelo modelo proposto.

3.3.5 Operações das classes de gerenciamento

Com o objetivo de promover a integridade do modelo de gerenciamento de customizações proposto seguindo as regras descritas no item 3.3.4, algumas operações podem ser incorporadas às classes de gerenciamento CProjeto, CElemento e CPropriedade. Estes métodos deverão efetuar o controle do gerenciamento das customizações e implementar as restrições no gerenciamento de customizações dos projetos, elementos e propriedades.

A Figura 30 apresenta o diagrama de classes de projetos com customização contendo as operações correspondentes a cada classe de gerenciamento.

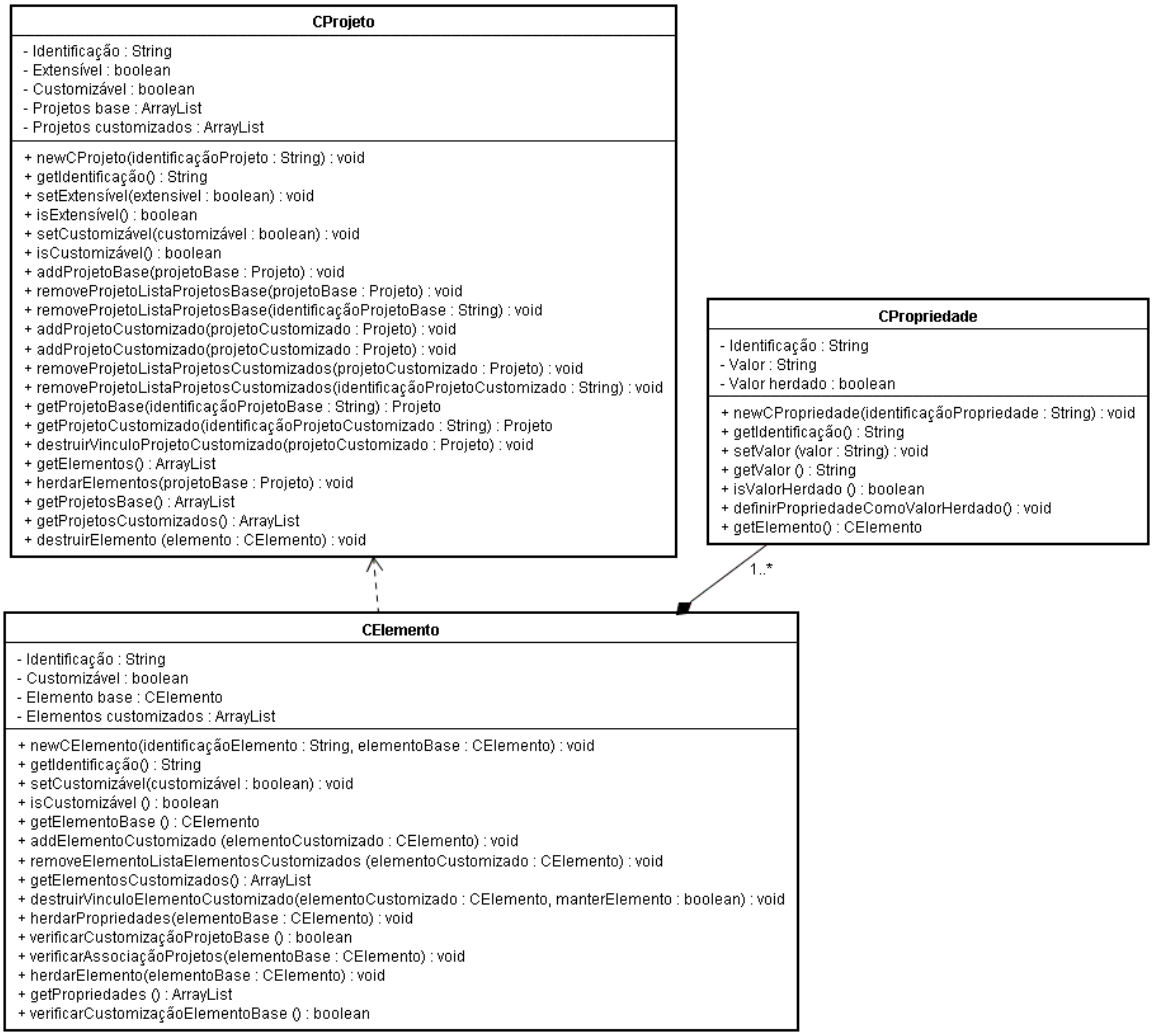

Figura 30 Diagrama de Classes de Projetos com Gerenciamento de Customização de Software

3.3.5.1 Operações da classe CProjeto

A seguir são descritas as operações da classe CProjeto:

newCProjeto (String identificaçãoProjeto)

Inicializa o projeto com o valor do atributo Identificação.

getIdentificação(): String

Retorna o valor do atributo Identificação de CProjeto.

setExtensível(boolean extensivel)

Define o valor do atributo Extensível de CProjeto.

isExtensível (): boolean

Retorna o valor do atributo Extensível de CProjeto.

setCustomizável (boolean customizável)

Define o valor do atributo Customizável de CProjeto.

isCustomizável (): boolean

Retorna o valor do atributo Customizável de CProjeto.

addProjetoBase (Projeto projetoBase)

Referência às Regras 1 e 2

Adiciona o projeto projetoBase na lista de “Projetos base” de CProjeto.

removeProjetoListaProjetosBase (Projeto projetoBase)

Remove o projeto da lista de “Projetos base” de CProjeto.

removeProjetoListaProjetosBase (String identificaçãoProjetoBase)

Remove o projeto com a identificaçãoProjetoBase da lista de “Projetos base” de CProjeto.

addProjetoCustomizado (Projeto projetoCustomizado)

Referência às Regras 2 e 8

Adiciona o projeto na lista de “Projetos customizados” de CProjeto.

addProjetoCustomizado (String identificaçãoProjetoCustomizado)

Referência às Regras 2 e 8

Cria um projeto customizado a partir do projeto e o adiciona à lista de “Projetos customizados”.

removeProjetoCustomizado (Projeto projetoCustomizado)

Remove o projeto da lista de “Projetos customizados” de CProjeto.

removeProjetoCustomizado (String identificaçãoProjetoCustomizado)

Remove o projeto com a identificaçãoProjetoCustomizado da lista de “Projetos customizados” de CProjeto.

getProjetoBase (String identificaçãoProjetoBase): Projeto

Retorna o projeto com a identificaçãoProjetoBase da lista de “Projetos base” de CProjeto.

getProjetoCustomizado (String identificaçãoProjetoCustomizado): Projeto

Retorna o projeto com a identificaçãoProjetoCustomizado da lista de “Projetos customizados” de CProjeto.

getProjetosBase (): Lista de Projetos

Retorna a lista de projetos “Projetos base” de CProjeto.

getProjetosCustomizados (): Lista de Projetos

Retorna a lista de projetos “Projetos customizados” de CProjeto.

destruirVinculoProjetoCustomizado (Projeto projetoCustomizado)

Referência às Regras 6 e 7

Destrói o vínculo do projeto com o projetoCustomizado (que utiliza este projeto como base). Torna os elementos de projetoCustomizado (herdados) independentes (próprios do projeto customizado).

getElementos (): Lista de objetos

Retorna a lista de todos os elementos do projeto.

herdarElementos (Projeto projetoBase)

Referência às Regras 4,8 e 12

Cria elementos para o projeto associados aos elementos de projetoBase quando este for associado ao projeto.

destruirElemento (Clemento elemento)

Referência à Regra 17

Destrói o elemento no projeto. Se existirem elementos customizados a partir deste elemento, estes elementos poderão ser destruídos ou o vínculo entre eles quebrado.

3.3.5.2 Operações da classe CElemento

A seguir são descritas as operações da classe CElemento:

newCElemento (String identificaçãoElemento, CElemento elementoBase)

Referência às Regra 9 e 16

Inicializa o elemento com o valor do atributo Identificação e o "Elemento base", se existir.

getIdentificação(): String

Retorna o valor do atributo Identificação de CElemento.

setCustomizável (boolean customizável)

Define o valor do atributo Customizável de CElemento.

isCustomizável (): boolean

Retorna o valor do atributo Customizável de CElemento.

getElementoBase (): CElemento

Retorna o "Elemento base" de CElemento.

addElementoCustomizado (CElemento elementoCustomizado)

Adiciona o elementoCustomizado à lista de "Elementos customizados" de CElemento.

removeElementoListaElementosCustomizados (CElemento elementoCustomziado)

Remove o elementoCustomizado da lista de "Elementos customizados" de CElemento.

getElementosCustomizados (): Lista de elementos

Retorna a lista de "Elementos customizados" de CElemento.

destruirVinculoElementoCustomizado (CElemento elementoCustomizado, boolean manterElemento)

Referência às Regras 13 e 14

Destrói o vínculo do elemento com o elementoCustomizado, que o utiliza como elemento base. Torna o elemento herdado independente (próprio do projeto customizado) se o parâmetro manterElemento for positivo e destrói o elemento herdado se o parâmetro manterElemento for negativo.

herdarPropriedades (CElemento elementoBase)

Referência às Regras 5,12 e 20

Cria as propriedades para o elemento (herdado) a partir das propriedades do elementoBase.

verificarCustomizaçãoProjetoBase (): boolean

Referência à Regra 3 e 8

Verificar a possibilidade de efetuar customizações no projeto customizado (elemento herdado do projeto base).

verificarAssociaçãoProjetos (CElemento elementoBase)

Referência à Regra 10

Verifica se o projeto do elementoBase está associado ao projeto do elemento como projeto base.

herdarElemento (CElemento elementoBase)

Referência às Regras 3 e 4

Constrói a associação do elemento com o elemento do projeto base.

getPropriedades (): Lista de CPropriedade

Retorna a lista das propriedades do elemento.

verificarCustomizaçãoElementoBase (): boolean

Referência à Regra 3 e 11

Verifica a possibilidade de efetuar customizações no elemento. Esta operação indica se o elemento herdado poderá realizar customizações em suas propriedades, de acordo com o valor do atributo Customizável do “Elemento base” deste elemento.

destruirPropriedadesElemento ()

Referência à Regra 17

Destrói as propriedades do elemento quando o elemento tiver que ser destruído.

3.3.5.3 Operações da classe CPropriedade

A seguir são descritas as operações da classe CPropriedade:

newCPropriedade (String identificaçãoPropriedade, boolean elementoHerdado)

Referência à Regra 20

Inicializa a propriedade com o valor do atributo Identificação. Se for uma propriedade herdada (propriredade criada a partir de uma propriedade de um elemento herdado – parâmetro elementoHerdado), atribui Sim ao atributo “Valor herdado” de CPropriedade. Do contrário, atribui Não ao atributo “Valor herdado” de CPropriedade.

getIdentificação(): String

Retorna o valor do atributo Identificação de CPropriedade.

setValor (String valor)

Define o valor do atributo Valor de CPropriedade.

getValor (): String

Referência às Regras 18,19 e 21

Retorna o valor do atributo Valor de CPropriedade. Se for um elemento herdado e o valor for herdado (atributo "Valor herdado"=Sim), busca o Valor de CPropriedade do elemento base.

isValorHerdado (): boolean

Retorna o valor do atributo "Valor herdado" de CPropriedade.

definirPropriedadeComoValorHerdado ()

Referência à Regra 22

Atribui o valor Sim ao atributo "Valor herdado" da propriedade.

getElemento (): CElemento

Retorna o elemento ao qual pertence esta propriedade.

3.3.6 Modelo de Informações para Projetos com Customização

No modelo de customização apresentado, a manutenção da herança entre os projetos é cada vez mais difícil à medida que aumenta a quantidade de níveis da hierarquia. Quando se tem um projeto customizado P(n) que herdou elementos de um projeto customizado P(n-1) que herdou elementos de um projeto customizado P(n-2) que herdou elementos de um projeto base P(n-3) a manutenção da herança pode prejudicar os projetos envolvidos. Nesta estrutura, o primeiro projeto mencionado (do software customizado P(n) está no 4º nível da hierarquia, e uma modificação realizada no software base P(n-3) deverá ser estendida a todos os projetos em níveis abaixo dele, exceto àqueles em que houve a customização.

O desenvolvimento de uma ferramenta que automatize o processo de gerenciamento das customizações irá facilitar a preservação da herança entre os diversos projetos, elementos e propriedades, bem como minimizar ou mesmo eliminar possíveis falhas na manutenção do modelo de customização em níveis.

Para auxiliar em uma possível implementação do gerenciamento de customizações de software, a Figura 31 apresenta uma tradução da proposta deste trabalho em um modelo lógico de dados.

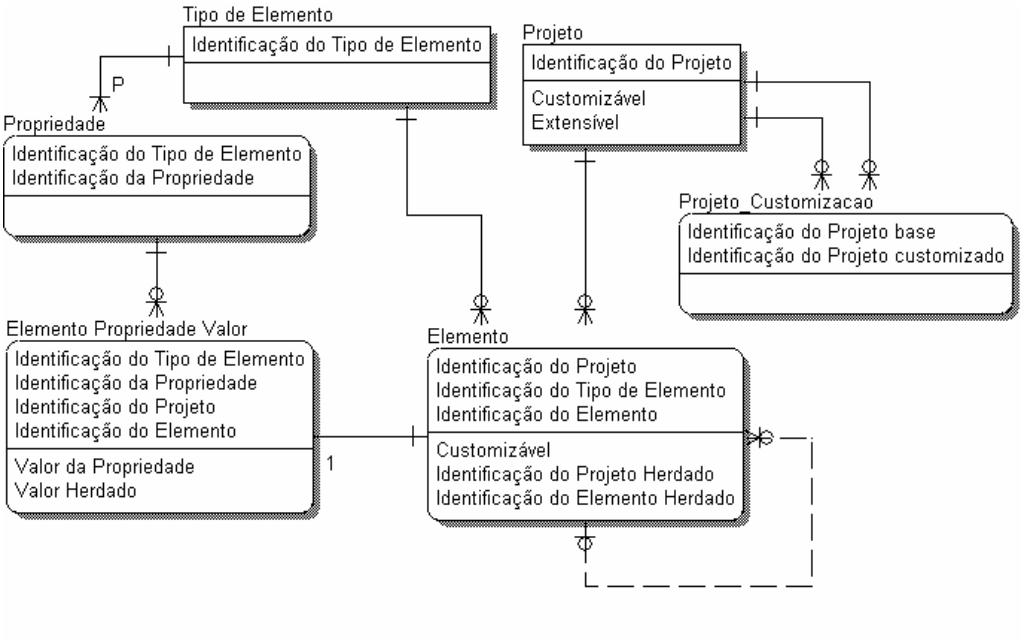

Figura 31 Modelo de Dados para o Gerenciamento de Customizações de Software

O modelo de dados apresentado (Figura 31) contém apenas as informações necessárias para a identificação dos projetos e de seus elementos. As informações complementares a cada entidade do modelo poderão ser definidas posteriormente, de acordo com a metodologia ou processo utilizado para desenvolvimento dos softwares com customização.

3.4 Conclusão

Neste capítulo foi apresentado um modelo de customização de softwares em níveis, onde são utilizados softwares como base para produção de novos softwares (customizados). Foi apresentada a proposta de um metamodelo para o gerenciamento de customizações de softwares, as regras para o modelo de gerenciamento de customizações de software e uma proposta de modelo de dados para a automatização desse processo de gerenciamento.

4 APLICAÇÃO DA PROPOSTA DE GERENCIAMENTO DE CUSTOMIZAÇÕES EM PROJETOS DE SOFTWARE EXEMPLO

A fim de demonstrar a prática do gerenciamento de customizações de software, segundo a proposta apresentada no Capítulo 3: seu funcionamento, a definição das propriedades de customização e observância das regras de gerenciamento, este capítulo aborda a aplicação do metamodelo de gerenciamento de customizações de software para o projeto de um software customizado. Para isso, serão utilizados alguns dos elementos de artefatos de software produzidos em algumas etapas do desenvolvimento de um software, bem como alguns softwares base.

4.1 OS TIPOS DE ELEMENTOS UTILIZADOS

Em virtude dos tipos de elementos e das propriedades de cada tipo de elemento do projeto ser definidas de acordo com o processo ou metodologia utilizada pela empresa desenvolvedora do software, foram definidos alguns tipos de elementos e algumas propriedades a serem tratados neste exemplo. Estes tipos de elementos e suas propriedades foram definidos com base na metodologia de desenvolvimento de software RUP, detalhada no capítulo 2.

Não foram extraídos da metodologia todos os tipos de elementos e as propriedades possíveis para cada um dos tipos. Como o objetivo do exemplo é demonstrar a aplicação do gerenciamento das customizações de software, serão utilizados apenas alguns tipos de elementos e apenas algumas propriedades desses elementos. O diagrama de classes da Figura 32 representa as classes que serão tratadas neste exemplo, incluindo as classes de gerenciamento de customizações. As propriedades dos tipos de elementos a serem utilizadas estão descritas na Tabela 4.

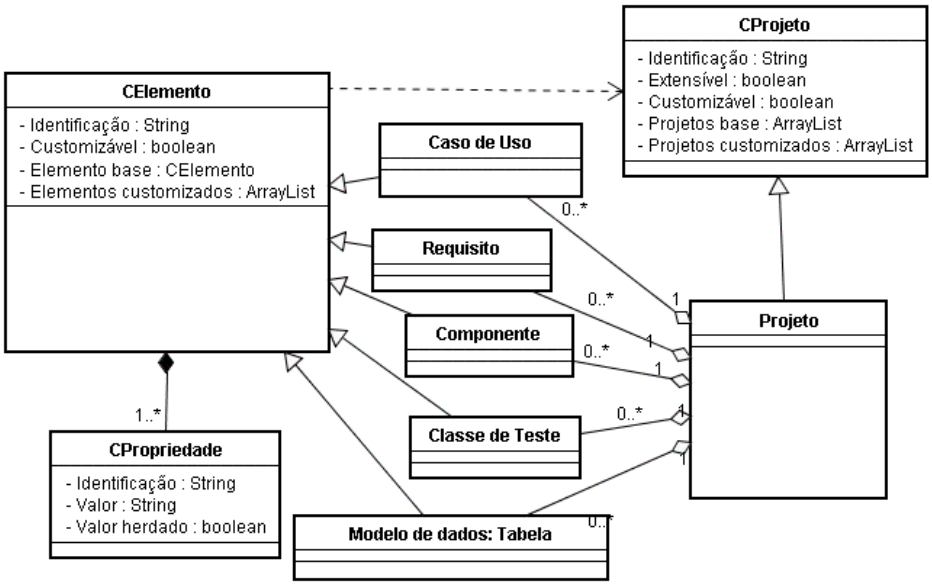

Figura 32 Diagrama de Classes dos Projetos com Customização do Estudo de Caso

Tabela 4

Tipos de Elementos e suas Propriedades

TIPO DE ELEMENTO	PROPRIEDADES
Caso de Uso	Nome, Breve Descrição
Requisito	Tipo, Descrição
Modelo de Dados: Tabela	Nome, Colunas
Componente	Nome, Corpo
Classe de Teste	Nome, Operações

4.2 O PROBLEMA

Uma empresa prestadora de serviços de *telemarketing* não dispõe de um sistema que opere seus contatos com os clientes. Seu controle é efetuado de forma manual onde existem fichas em papel para cada cartela de clientes das empresas que eles atendem. Ao contatar o cliente, o atendente anota os dados do contato na ficha correspondente.

4.3 SOFTWARES BASE

Para demonstrar a utilização de projetos de software base no desenvolvimento de software customizado, usaremos 3 projetos de software como base para o desenvolvimento de um software customizado que irá atender ao problema descrito no item 4.2:

- Cadastro de CEP
- Cadastro de Pessoas
- Autenticação de Usuários e Controle de Acesso

Os projetos de software base bem como alguns de seus elementos são descritos nos próximos itens deste documento. No Anexo I são apresentados e detalhados mais alguns elementos dos referidos projetos.

4.3.1 Cadastro de CEP

O software “Cadastro de CEP” é um sistema para inclusão, exclusão e alteração de CEP’s (Código de Endereçamento Postal). Alguns de seus principais elementos, produzidos durante todo o processo de desenvolvimento estão descritos nos próximos itens deste documento. A Figura 33 mostra o software “Cadastro de CEP” instanciado como objeto da classe Projeto.

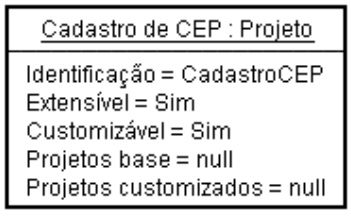

Figura 33 Objeto da classe Projeto (instância do projeto “Cadastro de CEP”)

4.3.1.1 Requisitos: Caso de Uso

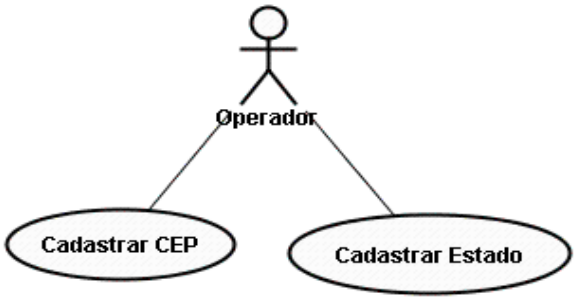

Figura 34 Modelo de Casos de Uso do software base “Cadastro de CEP”

A Figura 34 apresenta um modelo de casos de uso simplificado do software base “Cadastro de CEP”. Os casos de uso presentes no modelo são elementos do software base e terão as propriedades definidas para o tipo de elemento “Caso de Uso” (Tabela 4), bem como as propriedades de customização, definidas e descritas no Capítulo 3.

O Quadro 1 e o Quadro 2 mostram os elementos do tipo “Caso de Uso” e o mapeamento das propriedades desses elementos.

Quadro 1 Elemento: Caso de Uso – UC001 (Cadastro de CEP)

ELEMENTO: Caso de Uso – UC001		
Identificação	CadastroCEP_UC001	
Customizável	Sim	
Elemento base	*É Nulo*	
Elementos Customizados	*É Nulo*	
PROPRIEDADES DO ELEMENTO: Caso de Uso – UC001		
Identificação	**Valor**	**Valor herdado[4]**
Nome	Cadastrar CEP	Não
Breve Descrição	O Operador poderá cadastrar, alterar e excluir registros de CEP’s.	Não

Quadro 2 Elemento: Caso de Uso – UC002 (Cadastro de CEP)

ELEMENTO: Caso de Uso – UC002		
Identificação	CadastroCEP_UC002	
Customizável	Não	
Elemento base	*É Nulo*	
Elementos customizados	*É Nulo*	
PROPRIEDADES DO ELEMENTO: Caso de Uso – UC002		
Identificação	**Valor**	**Valor herdado**
Nome	Cadastrar Estado	Não
Breve Descrição	O Operador poderá cadastrar, alterar e excluir registros de Estados.	Não

4.3.2 Cadastro de Pessoas

O software "Cadastro de Pessoas" é um sistema para inclusão, exclusão e alteração de registros genéricos de pessoas, contendo algumas informações gerais sobre a pessoa. Alguns de seus principais elementos, produzidos durante todo o processo de desenvolvimento estão descritos nos próximos itens deste documento. A Figura 35 mostra o software "Cadastro de Pessoas" instanciado como objeto da classe Projeto.

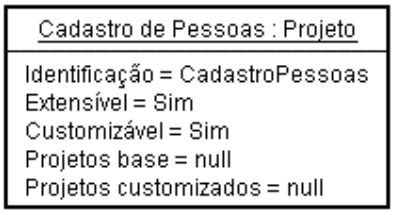

Figura 35 Objeto da classe Projeto (instância do projeto "Cadastro de Pessoas")

[4] O atributo "Valor herdado" de todas as propriedades dos elementos do projeto "Cadastro de CEP" estão definidos com valor "Não" porque seus elementos não foram herdados de nenhum projeto.

4.3.2.1 Requisitos: Caso de Uso

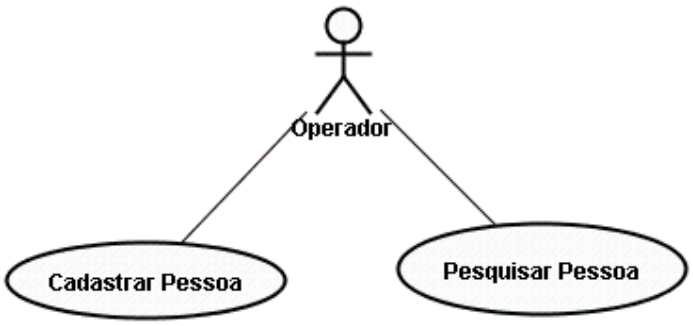

Figura 36 Modelo de Casos de Uso do software base “Cadastro de Pessoas”

A Figura 36 apresenta um modelo de casos de uso simplificado do software base “Cadastro de Pessoas”. Os casos de uso presentes no modelo são elementos do software base e terão as propriedades definidas para o tipo de elemento “Caso de Uso” (Tabela 4), bem como as propriedades de customização, definidas e descritas no Capítulo 3.

O Quadro 3 mostra o elemento do tipo “Caso de Uso” e o mapeamento das propriedades desse elemento.

Quadro 3 Elemento: Caso de Uso – UC001 (Cadastro de Pessoas)

ELEMENTO: Caso de Uso – UC001		
Identificação	CadastroPessoas_UC001	
Customizável	Sim	
Elemento base	*É Nulo*	
Elementos customizados	*É Nulo*	
PROPRIEDADES DO ELEMENTO: Caso de Uso – UC001		
Identificação	**Valor**	**Valor herdado**
Nome	Cadastrar Pessoa	Não[5]
Breve Descrição	O Operador poderá cadastrar, alterar e excluir registros de Pessoas.	Não

[5] O atributo “Valor herdado” de todas as propriedades dos elementos do projeto “Cadastro de Pessoas” está definido com valor “Não” porque seus elementos não foram herdados de nenhum projeto.

4.3.3 Autenticação de Usuários e Controle de Acesso

O software “Autenticação de Usuários e Controle de Acesso” é um sistema para inclusão, exclusão e alteração de usuários de sistemas e controle de acesso aos módulos de um sistema. Este software é um software customizado a partir de outro software, um software base. Por este motivo alguns dos elementos apresentados são herdados.

Neste exemplo, conforme já descrito, ele exerce o papel de um software base. Alguns de seus principais elementos, produzidos durante todo o processo de desenvolvimento estão descritos nos próximos itens deste documento. A Figura 37 mostra o software “Autenticação de Usuários e Controle de Acesso” instanciado como objeto da classe Projeto.

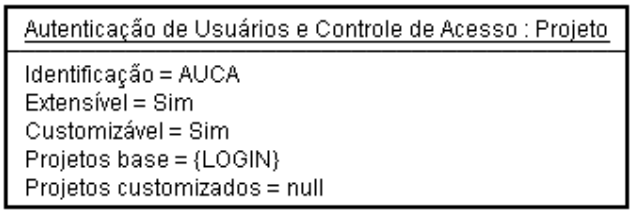

Figura 37 Objeto da classe Projeto (instância do projeto “Autenticação de Usuários e Controle de Acesso”)

4.3.3.1 Requisitos: Caso de Uso

Figura 38 Modelo de Casos de Uso do software base “Autenticação de Usuários e Controle de Acesso”

A Figura 38 apresenta um modelo de casos de uso simplificado do software base “Autenticação de Usuários e Controle de Acesso”. O caso de uso presente no modelo é um elemento do software base e terá as propriedades definidas para o tipo

de elemento “Caso de Uso” (Tabela 4), bem como as propriedades de customização, definidas e descritas no Capítulo 3.

O Quadro 4 mostra o elemento do tipo “Caso de Uso” e o mapeamento das propriedades desse elemento.

Quadro 4 Elemento: Caso de Uso – UC001 (Autenticação de Usuários e Controle de Acesso)

ELEMENTO: Caso de Uso – UC001		
*Elemento **herdado** do projeto **“LOGIN”***		
Identificação	AUCA_UC001	
Customizável	Sim	
Elemento base	(CElemento) projetoLOGIN_UC001	
Elementos customizados	*É Nulo*	
PROPRIEDADES DO ELEMENTO: Caso de Uso – UC003		
Identificação	**Valor**	**Valor Herdado**
Nome	Definir Acesso do usuário	Sim[6]
Breve Descrição	O Administrador poderá definir a quais os módulos determinados usuários terão acesso	Sim

4.4 Software customizado

No item 4.2 foi descrito o problema que deverá ser resolvido pelo software customizado. O software customizado deverá operar a administração do cadastro dos clientes, bem como controlar os contatos efetuados com os clientes.

Existem 3 projetos de software customizáveis que poderão ser base para este software customizado. Esses projetos base foram abordados no item 4.3 deste documento. Na Figura 39 está representada a hierarquia de customização dos projetos deste exemplo de aplicação. Ela apresenta o software customizado P(n), denominado CRM, em um nível abaixo dos softwares base P(n-1): “Cadastro de CEP”, “Cadastro de Pessoas” e “Autenticação de Usuário e Controle de Acesso”.

[6] O atributo “Valor herdado” de algumas propriedades dos elementos do projeto “Autenticação de Usuários e Controle de Acesso” está definido com valor “Sim” porque alguns elementos foram herdados do projeto LOGIN e não foram customizados (modificados).

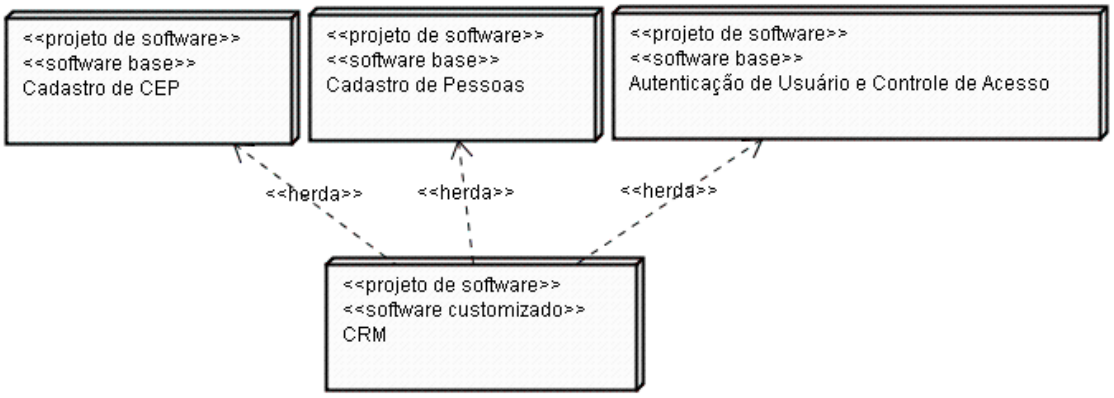

Figura 39 Diagrama de Utilização de Projetos com Customização para o Exemplo de Aplicação

4.4.1 CRM

Como os projetos de software “Cadastro de CEP”, “Cadastro de Pessoas” e “Autenticação de Usuário e Controle de Acesso” são base para o software customizado CRM, o projeto do software CRM herdará todos os elementos dos seus softwares base.

Para o exemplo serão descritos apenas alguns dos elementos produzidos durante o processo de desenvolvimento do software customizado CRM, incluindo os elementos herdados pelo processo de customização de software, ou seja, os elementos descritos nos itens 4.3.1 (Cadastro de CEP), 4.3.2 (Cadastro de Pessoas) e 4.3.3 (Autenticação de Usuário e Controle de Acesso) e Anexo I. Nos próximos itens são apresentados alguns dos elementos do projeto customizado. O detalhamento do gerenciamento de customizações dos demais elementos herdados pelo projeto customizado é apresentado no Anexo I deste trabalho.

A Figura 40 apresenta um diagrama de objetos que mostra o software customizado CRM instanciado como objeto da classe Projeto e relacionado aos objetos “Cadastro de CEP”, “Cadastro de Pessoas” e “Autenticação de usuário e Controle de Acesso”, também da classe Projeto.

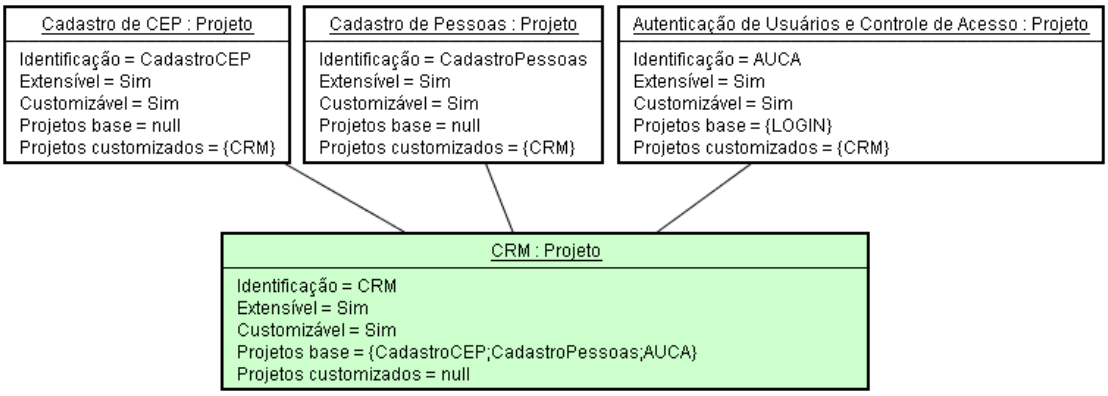

Figura 40 Diagrama de Objetos do projeto customizado CRM

4.4.1.1 Requisitos: Caso de Uso

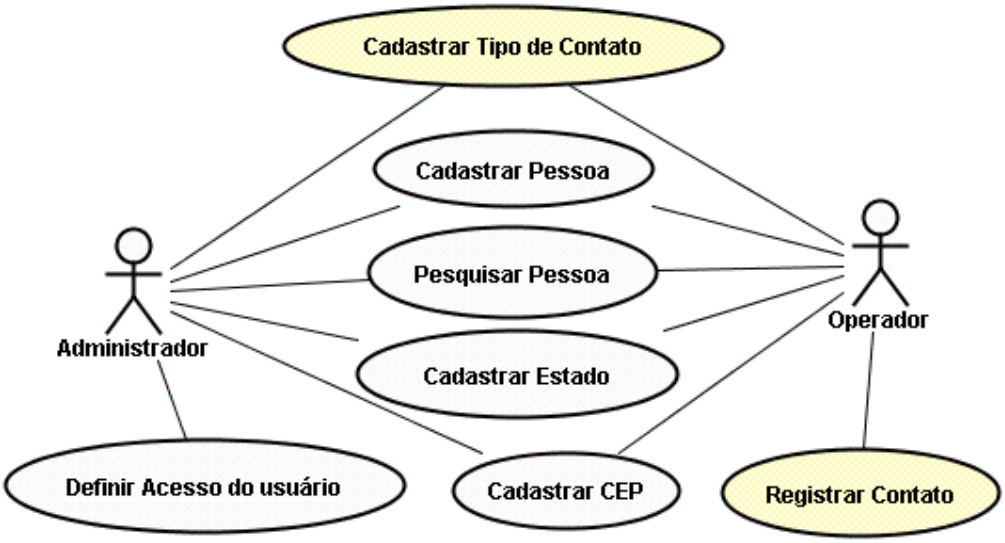

Figura 41 Modelo de Casos de Uso do software customizado CRM

A Figura 41 apresenta um modelo de casos de uso simplificado do software customizado CRM. Além dos casos de uso herdados dos softwares base, foram incluídos os casos de uso: “Cadastrar Tipo de Contato” e “Registrar Contato”, próprios do projeto do software customizado.

Os casos de uso presentes no modelo da Figura 41 são elementos do software customizado e terão as propriedades definidas para o tipo de elemento “Caso de Uso” (Tabela 4), bem como as propriedades de customização, definidas e descritas no Capítulo 3.

O Quadro 5 ao Quadro 9 mostram os elementos do tipo “Caso de Uso” e o mapeamento das propriedades desses elementos.

Quadro 5 Elemento: Caso de Uso – UC001 (CRM)

ELEMENTO: Caso de Uso – UC001			***Obs.***
*Elemento herdado do projeto “**Cadastro de CEP**”*			
Identificação	CRM_UC001		
Customizável	Sim		**1**
Elemento base	CadastroCEP_UC001		
Elementos customizados	*É Nulo*		
PROPRIEDADES DO ELEMENTO: Caso de Uso – UC001			
Nome	**Valor**	**Valor Herdado**	
Nome	Cadastrar CEP	Sim	**2**
Breve Descrição	O Operador poderá cadastrar, alterar e excluir registros de CEP’s.	Sim	**2**

Observações sobre o Quadro:

(numeração correspondente indicada no quadro, coluna Obs.)

Obs. 1 O elemento é customizável, ou seja, o atributo Customizável do elemento é definido com valor Sim. Isto indica que os valores das propriedades deste elemento (quando herdado) poderão ser alterados no projeto customizado. Como o atributo Customizável do elemento base deste elemento (CadastroCEP_UC001) também está definido com valor Sim, as propriedades deste elemento herdado (CRM_UC001) também poderão ser alteradas.

Obs. 2 Como este elemento foi herdado de outro projeto, o atributo “Valor herdado” das propriedades do elemento é definido com valor Sim a menos que o valor da propriedade venha a ser alterado. Neste caso, o valor do atributo “Valor herdado” deverá estar definido como Não.

Quadro 6 Elemento: Caso de Uso – UC002 (CRM)

ELEMENTO: Caso de Uso – UC002			***Obs.***
*Elemento herdado do projeto "**Cadastro de CEP**"*			
Identificação	CRM_UC002		
Customizável	Não		***1***
Elemento base	CadastroCEP_UC002		
Elementos customizados	*É Nulo*		
PROPRIEDADES DO ELEMENTO: Caso de Uso – UC002			
Nome	**Valor**	**Valor herdado**	
Nome	Cadastrar Estado	Sim	
Breve Descrição	O Operador poderá cadastrar, alterar e excluir registros de Estados.	Sim	

Observações sobre o Quadro:

(numeração correspondente indicada no quadro, coluna Obs.)

Obs. 1 Este elemento foi herdado e o elemento base está indicado como "não customizável", ou seja, valor da propriedade Customizável do elemento está definido com valor Não. Isto indica que os valores das propriedades deste elemento não poderão ser alterados. Quando o elemento foi herdado, a propriedade Customizável manteve seu valor original: Não. Neste caso, se este elemento for herdado por outro projeto, as propriedades do elemento herdado também não poderão ser alteradas no projeto customizado que o herdou.

Quadro 7 Elemento: Caso de Uso – UC003 (CRM)

ELEMENTO: Caso de Uso – UC003			***Obs.***
*Elemento herdado do projeto "**Cadastro de Pessoas**"*			
Identificação	CRM_UC003		
Customizável	Sim		
Elemento base	CadastroPessoas_UC001		
Elementos customizados	*É Nulo*		
PROPRIEDADES DO ELEMENTO: Caso de Uso – UC003			
Nome	**Valor**	**Valor Herdado**	
Nome	Cadastrar Pessoa	Sim	
Breve Descrição	O Operador poderá cadastrar, alterar e excluir registros de Pessoas.	Sim	

Quadro 8 Elemento: Caso de Uso – UC005 (CRM)

ELEMENTO: Caso de Uso – UC005			***Obs.***
*Elemento herdado do projeto "**Autenticação de Usuários e Controle de Acesso"***			
Identificação	CRM_UC005		
Customizável	Sim		
Elemento base	AUCA_UC001		***1***
Elementos customizados	*É Nulo*		***1***
PROPRIEDADES DO ELEMENTO: Caso de Uso – UC005			
Nome	**Valor**	**Valor herdado**	
Nome	Definir Acesso do usuário	Sim	
Breve Descrição	O Administrador poderá definir a quais os módulos determinados usuários terão acesso	Sim	

Observações sobre o Quadro:

(numeração correspondente indicada no quadro, coluna Obs.)

Obs. 1 Este elemento no software "Autenticação de usuário e Controle de Acesso" já havia sido herdado de outro software. No gerenciamento da customização de softwares as regras e a forma de mapear os atributos e propriedades de elementos herdados que foram herdados de outros projetos é a mesma, respeitando a hierarquia de níveis da customização de softwares.

Quadro 9 Elemento: Caso de Uso – UC006 (CRM)

ELEMENTO: Caso de Uso – UC006			***Obs.***
*Elemento **NÃO** foi herdado*			***1***
Identificação	CRM_UC006		
Customizável	Sim		
Elemento base	*É Nulo*		***1***
Elementos customizados	*É Nulo*		
PROPRIEDADES DO ELEMENTO: Caso de Uso – UC006			
Nome	**Valor**	**Valor herdado**	
Nome	Cadastrar Tipo de Contato	Não	***2***
Breve Descrição	O operador poderá inserir, alterar e excluir registros de tipos de contato	Não	***2***

Observações sobre o Quadro:

(numeração correspondente indicada no quadro, coluna Obs.)

Obs. 1 Este elemento é próprio do projeto de software customizado, não foi herdado de nenhum outro projeto.

Obs. 2 Como este elemento não foi herdado de outro projeto, o atributo “Valor herdado” das propriedades do elemento é definido com valor Não.

4.5 Conclusão

Neste capítulo foi apresentado um exemplo da aplicação do metamodelo de gerenciamento das customizações de um projeto de software (CRM) utilizando três projetos como base: “Cadastro de CEP”, “Cadastro de Pessoas” e “Autenticação de Usuário e Controle de Acesso”.

5 CONCLUSÕES E TRABALHOS FUTUROS

Pensar na customização de um software desde a sua concepção até sua implantação e manutenção pode trazer inúmeros benefícios. Dentre eles, a facilidade em gerenciar requisitos de um produto implantado em vários clientes ou mesmo a adaptação da interface a um usuário específico ou a perfis de usuários em uma mesma versão da aplicação. Com isso, modificações simples no processo de desenvolvimento do software e a adoção de um modelo de gerenciamento favorecem tanto o fornecedor, desenvolvedor do software, que poderá agilizar implantações e manutenções de seu produto quanto o cliente que terá uma resposta rápida ao atendimento de suas necessidades.

5.1 SUGESTÕES DE TRABALHOS FUTUROS

Como a proposta deste trabalho era a de desenvolver um modelo de gerenciamento de customizações de software, não uma ferramenta de gerenciamento, como trabalhos futuros, sugere-se a automatização do metamodelo de gerenciamento de customizações proposto na forma de uma ferramenta que auxilie na criação dos projetos/elementos e gerenciamento de sua customização.

Além da automatização do metamodelo de gerenciamento de customizações de software, algumas questões não foram cobertas pela proposta deste trabalho:

- Extensão do metamodelo de gerenciamento de customizações de software para atender as situações em que um elemento do projeto dependa ou esteja relacionado a outro elemento do mesmo projeto, como no caso Tabela->Coluna, por exemplo. Uma vez que tipos de elementos podem ser relacionados entre si, como no caso Tabela->Coluna ou mesmo "Modelo de Casos de Uso"->"Caso de Uso", a adaptação da proposta deste trabalho para que este relacionamento exista, com a devida automatização e implementação, poderá possibilitar a geração automatizada de artefatos de software para o projeto.
- Proposta de uma nomenclatura a ser utilizada para projetos, elementos e propriedades no sentido de evitar possíveis conflitos de "identificação" destes componentes em projetos customizados. Uma vez que haja a automatização do processo de gerenciamento de customizações de software, a utilização de projetos base que contenham elementos com a mesma identificação (mesmo

valor do atributo identificação de elemento) poderá acarretar em conflitos na identificação destes elementos em projetos customizados. Se houver uma padronização quanto à identificação de projetos e elementos, este problema poderia ser amenizado ou mesmo eliminado.

- Extensão da proposta de gerenciamento de customizações de software para permitir a administração e o registro da evolução histórica das diferentes versões de cada elemento ou artefato.

5.2 CONSIDERAÇÕES FINAIS

A prática da customização de software pode ser adotada há bastante tempo nas empresas que desenvolvem software, no entanto a literatura que descreve e discute essa atividade ainda é muito escassa. Foi observado ao longo do desenvolvimento deste trabalho que os conceitos de reusabilidade e manutenibilidade, boas práticas na produção de software, estão fortemente relacionados à customização de software, uma vez que o reuso de artefatos e a previsão e qualidade de manutenções são inerentes ao processo de customização.

Embora a customização de um software e o uso de projetos de software como base possa ser aplicado a diversos projetos e em diversos níveis, existem casos em que o nível de especialização do projeto customizado o transforma em um projeto que não poderá ser estendido. Quando um software atinge um nível de customização muito grande, ele pode tornar-se muito específico a um cliente ou tipo de cliente. Ou seja, não poderá ser utilizado como base para outros projetos customizados.

Outra questão é a aplicação do modelo de gerenciamento de customizações de software de forma manual. É possível, mas não viável. A quantidade de elementos, particularidades e regras de gerenciamento tornam a aplicação do modelo inviável se não for automatizada, uma vez que um único erro poderá comprometer toda a estrutura de projetos customizados.

Entretanto, mesmo com a existência de dificultadores, inerentes a aplicação de uma nova forma de se desenvolver software, a customização e o gerenciamento de customizações de software têm o potencial para ser uma importante contribuição à relação qualidade, tempo e custo, tão discutida atualmente pelos que fornecem e pelos que adquirem software.

REFERÊNCIAS

ARAUJO, Vânia Maria Rodrigues Hermes de. **Sistemas de informação: nova abordagem teórico-conceitual**. Ciência da Informação, Ibict, Brasília, v. 24, n. 1, 1995.
Disponível em: <http://www.ibict.br/cienciadainformacao/>. Acesso em: 14 jun. 2008.

BRUSAMOLIN, Valério. **MANUTENIBILIDADE DE SOFTWARE**. Revista Digital: Comunicações Digitais e Tópicos Relacionados, Icesp, v. 2, n., p.10-15, 20 jan. 2004.
Disponível em: <http://www.revdigonline.com>. Acesso em: 20 jan. 2004.

KRUCHTEN, Philippe. **Introdução ao RUP – *Rational Unified Process***. 2ª Edição, Rio de Janeiro: Editora Ciência Moderna Ltda., 2003. 255p.

MICHAELIS: Moderno Dicionário da Língua Portuguesa.
Disponível em: <http://michaelis.uol.com.br/moderno/portugues/index.php?>. Acesso em: 06 set. 2008.

NOMURA, Luzia; SPINOLA, Mauro. **Processo de manutenção de software: um modelo de melhoria da qualidade com ênfase na manutenção preventiva**. 120 p. Tese (Mestrado) – Universidade paulista, São Paulo, 2001.

OLIVEIRA, Djalma de Pinho Rebouças. **Sistemas de Informações Gerenciais**. 4. ed. São Paulo: Atlas, 1997.

PIACENTINI, Maria Tereza de Queiroz. **Customizar, fulcrar, ociar**.
Disponível em:
<http://kplus.cosmo.com.br/materia.asp?co=148&rv=Gramatica>.
Acesso em: 14 jun. 2008

PRESSMAN, Roger S. **Engenharia de Software**. São Paulo: Makron Books, 1995.

RATIONAL Software Corporation. **Rational Unified Process**. Versão 2002.05.00 Rational, 2002. CD-ROM.

REZENDE, Denis Alcides. **Engenharia de Software e Sistemas de Informação**. Rio de Janeiro: Brasport, 1999. 292 p.

ROSELINO, José Eduardo. **A INDÚSTRIA DE SOFTWARE: o "modelo brasileiro" em perspectiva comparada**. 2006. 236 f. Tese (Doutorado) - Curso de Ciências Econômicas, Departamento de Instituto de Economia, Universidade Estadual de Campinas, São Paulo, 2006.

SILVA, Ricardo Pereira e. **Suporte ao desenvolvimento e uso de frameworks e componentes**. 2000. 262 f. Tese (Doutorado) - Departamento de Ppgc, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2000.

SILVA, Ricardo Pereira. **UML 2:** Modelagem Orientada a Objetos. Florianópolis: Visual Books, 2007. 232 p.

6 ANEXO I

Este anexo apresenta o detalhamento dos elementos dos projetos de software “Cadastro de CEP”, “Cadastro de Pessoas”, “Autenticação de Usuários e Controle de Acesso” e CRM, apresentados no capítulo 4.

6.1 SOFTWARES BASE

6.1.1 Cadastro de CEP

6.1.1.1 Requisitos: Requisito de Software

Os requisitos de software presentes no software base “Cadastro de CEP” terão as propriedades definidas para o tipo de elemento Requisito (Tabela 4), bem como as propriedades de customização, definidas e descritas no Capítulo 3.

O Quadro 10 ao Quadro 12 mostram os elementos do tipo Requisito e o mapeamento das propriedades desses elementos.

Quadro 10 Elemento: Requisito – RF001 (Cadastro de CEP)

ELEMENTO: Requisito – RF001		
Identificação	CadastroCEP_RF001	
Customizável	Sim	
Elemento base	*É Nulo*	
Elementos customizados	*É Nulo*	
PROPRIEDADES DO ELEMENTO: Requisito – RF001		
Identificação	**Valor**	**Valor herdado**
Tipo	Funcional	Não
Descrição	O Operador poderá cadastrar, alterar e excluir registros de CEP’s.	Não

Quadro 11 Elemento: Requisito – RF002 (Cadastro de CEP)

ELEMENTO: Requisito – RF002		
Identificação	CadastroCEP_RF002	
Customizável	Sim	
Elemento base	*É Nulo*	
Elementos customizados	*É Nulo*	
PROPRIEDADES DO ELEMENTO: Requisito – RF002		
Identificação	**Valor**	**Valor herdado**
Tipo	Funcional	Não
Descrição	O Operador poderá pesquisar Municípios pelo nome e pelo Estado.	Não

Quadro 12 Elemento: Requisito – RF003 (“Cadastro de CEP”)

ELEMENTO: Requisito – RF003		
Identificação	CadastroCEP_RF003	
Customizável	Sim	
Elemento base	*É Nulo*	
Elementos customizados	*É Nulo*	
PROPRIEDADES DO ELEMENTO: Requisito – RF003		
Identificação	**Valor**	**Valor herdado**
Tipo	Funcional	Não
Descrição	O Operador poderá pesquisar CEP’s pelo número do CEP, pelo Município, pelo Bairro e pelo logradouro.	Não

6.1.1.2 Análise e Projeto: Modelo de Dados: Tabela

As tabelas presentes no software base “Cadastro de CEP” terão as propriedades definidas para o tipo de elemento “Modelo de Dados: Tabela” (Tabela 4), bem como as propriedades de customização, definidas e descritas no Capítulo 3.

O Quadro 13 mostra o elemento do tipo “Modelo de Dados: Tabela” e o mapeamento das propriedades desse elemento.

Quadro 13 Elemento: Modelo de Dados: Tabela – LOGRADOURO (Cadastro de CEP)

ELEMENTO: Modelo de Dados: Tabela – LOGRADOURO		
Identificação	CadastroCEP_logradouro	
Customizável	Sim	
Elemento base	*É Nulo*	
Elementos customizados	*É Nulo*	
PROPRIEDADES DO ELEMENTO: Modelo de Dados: Tabela – LOGRADOURO		
Identificação	**Valor**	**Valor herdado**
Nome	CEP_LOGRADOURO	Não
Colunas	COD_LOGRADOURO;CEP;LOGRADOURO;BAIRRO;COD_ESTADO;COD_MUNICIPIO	Não

6.1.1.3 Implementação: Componente do Software

Os componentes de software presentes no software base “Cadastro de CEP” terão as propriedades definidas para o tipo de elemento “Componente” (Tabela 4), bem como as propriedades de customização definidas e descritas no Capítulo 3.

O Quadro 14 mostra o elemento do tipo “Componente” e o mapeamento das propriedades desse elemento.

Quadro 14 Elemento: Componente – controleLOGRADOURO (Cadastro de CEP)

ELEMENTO: Componente – controleLOGRADOURO		
Identificação	CadastroCEP_controleLogradouro	
Customizável	Sim	
Elemento base	*É Nulo*	
Elementos customizados	*É Nulo*	
PROPRIEDADES DO ELEMENTO: Componente – controleLOGRADOURO		
Identificação	**Valor**	**Valor herdado**
Nome	controleLogradouro.java	Não
Corpo	.../*corpo da classe controleLogradouro.java*/...	Não

6.1.1.4 Testes: Classe de Teste

As classes de teste presentes no software base “Cadastro de CEP” terão as propriedades definidas para o tipo de elemento “Classe de Teste” (Tabela 4), bem como as propriedades de customização, definidas e descritas no Capítulo 3.

O Quadro 15 mostra o elemento do tipo “Classe de Teste” e o mapeamento das propriedades desse elemento.

Quadro 15 Elemento: Classe de Teste – tControleLOGRADOURO (Cadastro de CEP)

ELEMENTO: Classe de Teste – tControleLOGRADOURO		
Identificação	CadastroCEP_tControleLogradouro	
Customizável	Sim	
Elemento base	*É Nulo*	
Elementos customizados	*É Nulo*	
PROPRIEDADES DO ELEMENTO: Classe de Teste – tControleLOGRADOURO		
Identificação	**Valor**	**Valor herdado**
Nome	tControleLogradouro.java	Não
Operações	tInsertLogradouro (Logradouro newLogradouro);tDeleteLogradouro (int codLogradouro);tUpdateLogradouro (Logradouro newLogradouro, Logradouro oldLogradouro);	Não

6.1.2 Cadastro de Pessoas

6.1.2.1 Requisitos: Caso de Uso

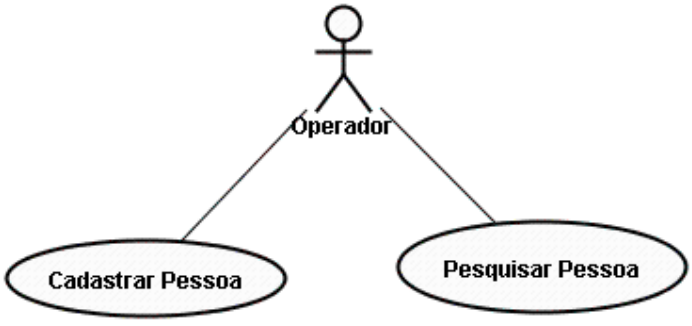

Figura 42 Modelo de Casos de Uso do software base “Cadastro de Pessoas”

A Figura 36 apresenta um modelo de casos de uso simplificado do software base “Cadastro de Pessoas”. Os casos de uso presentes no modelo são elementos do software base e terão as propriedades definidas para o tipo de elemento “Caso de Uso” (Tabela 4), bem como as propriedades de customização, definidas e descritas no Capítulo 3.

O Quadro 16 mostra o elemento do tipo “Caso de Uso” e o mapeamento das propriedades desse elemento.

Quadro 16 Elemento: Caso de Uso – UC002 (Cadastro de Pessoas)

ELEMENTO: Caso de Uso – UC002		
Identificação	CadastroPessoas_UC002	
Customizável	Sim	
Elemento base	*É Nulo*	
Elementos customizados	*É Nulo*	
PROPRIEDADES DO ELEMENTO: Caso de Uso – UC002		
Identificação	**Valor**	**Valor herdado**
Nome	Pesquisar Pessoa	Não
Breve Descrição	O Operador pesquisar as pessoas cadastradas	Não

6.1.2.2 Requisitos: Requisito de Software

Os requisitos de software presentes no software base “Cadastro de Pessoas” terão as propriedades definidas para o tipo de elemento Requisito (Tabela 4), bem como as propriedades de customização, definidas e descritas no Capítulo 3.

O Quadro 17 mostra o elemento do tipo Requisito e o mapeamento das propriedades desse elemento.

Quadro 17 Elemento: Requisito – RF001 (Cadastro de Pessoas)

ELEMENTO: Requisito – RF001		
Identificação	CadastroPessoas_RF001	
Customizável	Sim	
Elemento base	*É Nulo*	
Elementos customizados	*É Nulo*	
PROPRIEDADES DO ELEMENTO: Requisito – RF001		
Identificação	**Valor**	**Valor herdado**
Tipo	Funcional	Não
Descrição	O Operador poderá pesquisar as pessoas cadastradas pelo nome e pelo CPF	Não

6.1.2.3 Análise e Projeto: Modelo de Dados: Tabela

As tabelas presentes no software base “Cadastro de Pessoas” terão as propriedades definidas para o tipo de elemento “Modelo de Dados: Tabela” (Tabela 4), bem como as propriedades de customização, definidas e descritas no Capítulo 3.

O Quadro 18 mostra o elemento do tipo “Modelo de Dados: Tabela” e o mapeamento das propriedades desse elemento.

Quadro 18 Elemento: Modelo de Dados: Tabela – PESSOA (Cadastro de Pessoas)

ELEMENTO: Modelo de Dados: Tabela – PESSOA		
Identificação	CadastroPessoas_pessoa	
Customizável	Sim	
Elemento base	*É Nulo*	
Elementos customizados	*É Nulo*	
PROPRIEDADES DO ELEMENTO: Modelo de Dados: Tabela – PESSOA		
Identificação	**Valor**	**Valor herdado**
Nome	PESSOA	Não
Colunas	COD_PESSOA;NOME;CPF;DT_NASC;ENDERECO;TELEFONE;EMAIL	Não

6.1.2.4 Implementação: Componente do Software

Os componentes de software presentes no software base “Cadastro de Pessoas” terão as propriedades definidas para o tipo de elemento “Componente” (Tabela 4), bem como as propriedades de customização definidas e descritas no Capítulo 3.

O Quadro 19 mostra o elemento do tipo “Componente” e o mapeamento das propriedades desse elemento.

Quadro 19 Elemento: Componente – controlePESSOA (Cadastro de Pessoas)

ELEMENTO: Componente – controlePESSOA		
Identificação	CadastroPessoas_controlePessoa	
Customizável	Sim	
Elemento base	*É Nulo*	
Elementos customizados	*É Nulo*	
PROPRIEDADES DO ELEMENTO: Componente – controlePESSOA		
Identificação	**Valor**	**Valor herdado**
Nome	controlePessoa.java	Não
Corpo	.../*corpo da classe controlePessoa.java*/...	Não

6.1.3 Autenticação de Usuários e Controle de Acesso

6.1.3.1 Requisitos: Requisito de Software

Os requisitos de software presentes no software base “Autenticação de Usuários e Controle de Acesso” terão as propriedades definidas para o tipo de elemento Requisito (Tabela 4), bem como as propriedades de customização, definidas e descritas no Capítulo 3.

O Quadro 20 e Quadro 21 mostram os elementos do tipo Requisito e o mapeamento das propriedades desses elementos.

Quadro 20 Elemento: Requisito – RNF001 (Autenticação de Usuários e Controle de Acesso)

ELEMENTO: Requisito – RNF001		
*Elemento **herdado** do projeto **"LOGIN"***		
Identificação	AUCA_RNF001	
Customizável	Sim	
Elemento base	(CElemento) projetoLOGIN_RNF004	
Elementos customizados	*É Nulo*	
PROPRIEDADES DO ELEMENTO: Requisito – RNF001		
Identificação	**Valor**	**Valor Herdado**
Tipo	Não Funcional	Sim
Descrição	Apenas usuários devidamente autenticados e do tipo administrador poderão acessar o sistema	Sim

Quadro 21 Elemento: Requisito – RNF002 (Autenticação de Usuários e Controle de Acesso)

ELEMENTO: Requisito – RNF002		
Identificação	AUCA_RNF002	
Customizável	Sim	
Elemento base	*É Nulo*	
Elementos customizados	*É Nulo*	
PROPRIEDADES DO ELEMENTO: Requisito – RNF002		
Identificação	**Valor**	**Valor herdado**
Tipo	Não Funcional	Não
Descrição	Apenas os módulos indicados como restritos deverão ter seu acesso mapeado aos usuários. Módulos não restritos podem ser acessados por todos os usuários do sistema	Não

6.1.3.2 Análise e Projeto: Modelo de Dados: Tabela

As tabelas presentes no software base "Autenticação de Usuários e Controle de Acesso" terão as propriedades definidas para o tipo de elemento "Modelo de Dados: Tabela" (Tabela 4), bem como as propriedades de customização, definidas e descritas no Capítulo 3.

O Quadro 22 e Quadro 23 mostram os elementos do tipo "Modelo de Dados: Tabela" e o mapeamento das propriedades desses elementos.

Quadro 22 Elemento: Modelo de Dados: Tabela – MODULO (Autenticação de Usuários e Controle de Acesso)

ELEMENTO: Modelo de Dados: Tabela – MODULO		
Identificação	AUCA_modulo	
Customizável	Não	
Elemento base	*É Nulo*	
Elementos customizados	*É Nulo*	
PROPRIEDADES DO ELEMENTO: Modelo de Dados: Tabela – MODULO		
Identificação	**Valor**	**Valor herdado**
Nome	LOGIN_MODULO	Não
Colunas	COD_MODULO;MODULO;DESCRICAO;ACESSO_RESTRITO	Não

Quadro 23 Elemento: Modelo de Dados: Tabela – ACESSO (Autenticação de Usuários e Controle de Acesso)

ELEMENTO: Modelo de Dados: Tabela – ACESSO		
Identificação	AUCA_acesso	
Customizável	Não	
Elemento base	*É Nulo*	
Elementos customizados	*É Nulo*	
PROPRIEDADES DO ELEMENTO: Modelo de Dados: Tabela – ACESSO		
Identificação	**Valor**	**Valor herdado**
Nome	LOGIN_USUARIO_ACESSO	Não
Colunas	COD_USUARIO;COD_MODULO	Não

6.1.3.3 Testes: Classe de Teste

As classes de teste presentes no software base “Autenticação de Usuários e Controle de Acesso” terão as propriedades definidas para o tipo de elemento “Classe de Teste” (Tabela 4), bem como as propriedades de customização, definidas e descritas no Capítulo 3.

O Quadro 24 mostra o elemento do tipo “Classe de Teste” e o mapeamento das propriedades desse elemento.

Quadro 24 Elemento: Classe de Teste – tControleLOGIN (Autenticação de Usuários e Controle de Acesso)

ELEMENTO: Classe de Teste – tControleLOGIN		
*Elemento **herdado** do projeto **"LOGIN"***		
Identificação	AUCA_tControleLOGIN	
Customizável	Sim	
Elemento base	(CElemento) projetoLOGIN_tControleLOGIN	
Elementos customizados	*É Nulo*	
PROPRIEDADES DO ELEMENTO: Classe de Teste – tControleLOGIN		
Identificação	**Valor**	**Valor herdado**
Nome	tControleLogin.java	Sim
Operações	tLoginUsuario (String usuario, String senha); tTrocaSenha (String usuario, String senhaAtual, String novaSenha)	Não

6.2 Software customizado

6.2.1 CRM

6.2.1.1 Requisitos: Caso de Uso

O Quadro 25 ao Quadro 29 mostram os elementos do tipo "Caso de Uso" e o mapeamento das propriedades desses elementos.

Quadro 25 Elemento: Caso de Uso – UC003 (CRM)

ELEMENTO: Caso de Uso – UC003			***Obs.***
*Elemento herdado do projeto "**Cadastro de Pessoas**"*			
Identificação	CRM_UC003		
Customizável	Sim		
Elemento base	CadastroPessoas_UC001		
Elementos customizados	*É Nulo*		
PROPRIEDADES DO ELEMENTO: Caso de Uso – UC003			
Nome	**Valor**	**Valor Herdado**	
Nome	Cadastrar Pessoa	Sim	
Breve Descrição	O Operador poderá cadastrar, alterar e excluir registros de Pessoas.	Sim	

Quadro 26 Elemento: Caso de Uso – UC004 (CRM)

ELEMENTO: Caso de Uso – UC004			***Obs.***
*Elemento herdado do projeto "**Cadastro de Pessoas**"*			
Identificação	CRM_UC004		
Customizável	Sim		
Elemento base	CadastroPessoas_UC002		
Elementos customizados	*É Nulo*		
PROPRIEDADES DO ELEMENTO: Caso de Uso – UC004			
Nome	**Valor**	**Valor Herdado**	
Nome	Pesquisar Pessoa	Sim	
Breve Descrição	O Operador pesquisar as pessoas cadastradas	Sim	

Quadro 27 Elemento: Caso de Uso – UC005 (CRM)

ELEMENTO: Caso de Uso – UC005			***Obs.***
*Elemento herdado do projeto "**Autenticação de Usuários e Controle de Acesso**"*			
Identificação	CRM_UC005		
Customizável	Sim		
Elemento base	AUCA_UC001		***1***
Elementos customizados	*É Nulo*		***1***
PROPRIEDADES DO ELEMENTO: Caso de Uso – UC005			
Nome	**Valor**	**Valor herdado**	
Nome	Definir Acesso do usuário	Sim	
Breve Descrição	O Administrador poderá definir a quais os módulos determinados usuários terão acesso	Sim	

Observações sobre o Quadro:

(numeração correspondente indicada no quadro, coluna Obs.)

Obs. 1 Este elemento no software “Autenticação de usuário e Controle de Acesso” já havia sido herdado de outro software. No gerenciamento da customização de softwares as regras e a forma de mapear os atributos e propriedades de elementos herdados que foram herdados de outros projetos é a mesma, respeitando a hierarquia de níveis da customização de softwares.

Quadro 28 Elemento: Caso de Uso – UC006 (CRM)

ELEMENTO: Caso de Uso – UC006			***Obs.***
*Elemento **NÃO** foi herdado*			***1***
Identificação	CRM_UC006		
Customizável	Sim		
Elemento base	*É Nulo*		***1***
Elementos customizados	*É Nulo*		
PROPRIEDADES DO ELEMENTO: Caso de Uso – UC006			
Nome	**Valor**	**Valor herdado**	
Nome	Cadastrar Tipo de Contato	Não	***2***
Breve Descrição	O operador poderá inserir, alterar e excluir registros de tipos de contato	Não	***2***

Observações sobre o Quadro:

(numeração correspondente indicada no quadro, coluna Obs.)

Obs. 1 Este elemento é próprio do projeto de software customizado, não foi herdado de nenhum outro projeto.

Obs. 2 Como este elemento não foi herdado de outro projeto, o atributo “Valor herdado” das propriedades do elemento é definido com valor Não.

Quadro 29 Elemento: Caso de Uso – UC007 (CRM)

ELEMENTO: Caso de Uso – UC007			***Obs.***
*Elemento **NÃO** foi herdado*			
Identificação	CRM_UC007		
Customizável	Sim		
Elemento base	*É Nulo*		
Elementos customizados	*É Nulo*		
PROPRIEDADES DO ELEMENTO: Caso de Uso – UC007			
Nome	**Valor**	**Valor herdado**	
Nome	Registrar Contato	Não	
Breve Descrição	O operador poderá registrar contatos efetuados com o cliente	Não	

6.2.1.2 Requisitos: Requisito de Software

Os requisitos de software presentes no software customizado CRM terão as propriedades definidas para o tipo de elemento Requisito (Tabela 4), bem como as propriedades de customização, definidas e descritas no Capítulo 3.

O Quadro 30 ao Quadro 37 mostram os elementos do tipo Requisito e o mapeamento das propriedades desses elementos.

Quadro 30 Elemento: Requisito – RF001 (CRM)

ELEMENTO: Requisito – RF001			***Obs.***
*Elemento herdado do projeto "**Cadastro de CEP**"*			
Identificação	CRM_RF001		
Customizável	Sim		
Elemento base	CadastroCEP_RF001		
Elementos customizados	*É Nulo*		
PROPRIEDADES DO ELEMENTO: Requisito – RF001			
Nome	**Valor**	**Valor herdado**	
Tipo	Funcional	Sim	
Descrição	O Operador poderá cadastrar, alterar e excluir registros de CEP's.	Sim	

Quadro 31 Elemento: Requisito – RF002 (CRM)

ELEMENTO: Requisito – RF002			***Obs.***
*Elemento herdado do projeto "**Cadastro de CEP**"*			
Identificação	CRM_RF002		
Customizável	Sim		
Elemento base	CadastroCEP_RF002		
Elementos customizados	*É Nulo*		
PROPRIEDADES DO ELEMENTO: Requisito – RF002			
Nome	**Valor**	**Valor herdado**	
Tipo	Funcional	Sim	
Descrição	O Operador poderá pesquisar Municípios pelo nome e pelo Estado.	Sim	

Quadro 32 Elemento: Requisito – RF003 (CRM)

ELEMENTO: Requisito – RF003			***Obs.***
*Elemento herdado do projeto "**Cadastro de CEP**"*			
Identificação	CRM_RF003		
Customizável	Sim		
Elemento base	CadastroCEP_RF003		
Elementos customizados	*É Nulo*		
PROPRIEDADES DO ELEMENTO: Requisito – RF003			
Nome	**Valor**	**Valor Herdado**	
Tipo	Funcional	Sim	
Descrição	O Operador poderá pesquisar CEP's pelo número do CEP, pelo Município, pelo Bairro e pelo logradouro.	Sim	

Quadro 33 Elemento: Requisito – RF004 (CRM)

ELEMENTO: Requisito – RF004			**Obs.**
*Elemento herdado do projeto "**Cadastro de Pessoas**"*			
Identificação	CRM_RF004		
Customizável	Sim		
Elemento base	CadastroPessoas_RF001		
Elementos customizados	*É Nulo*		
PROPRIEDADES DO ELEMENTO: Requisito – RF004			
Nome	**Valor**	**Valor herdado**	
Tipo	Funcional	Sim	
Descrição	O Operador poderá pesquisar as pessoas cadastradas pelo nome, pelo CPF, pelo email e pelo CEP	Não	**1**

Observações sobre o Quadro:

(numeração correspondente indicada no quadro, coluna Obs.)

Obs. 1 Este elemento foi herdado de outro projeto (Cadastro de Pessoas) e, pelas regras o valor do atributo "Valor herdado" deveria ser Sim. No entanto, como houve customização (alteração) do valor da propriedade Descrição, o valor do atributo "Valor herdado" passa a ser Não.

Quadro 34 Elemento: Requisito – RF005 (CRM)

ELEMENTO: Requisito – RF005			***Obs.***
*Elemento **NÃO** foi herdado*			
Identificação	CRM_RF005		
Customizável	Sim		
Elemento base	*É Nulo*		
Elementos customizados	*É Nulo*		
PROPRIEDADES DO ELEMENTO: Requisito – RF005			
Nome	**Valor**	**Valor herdado**	
Tipo	Funcional	Não	
Descrição	O operador poderá registrar contatos efetuados com os clientes.	Não	

Quadro 35 Elemento: Requisito – RNF001 (CRM)

ELEMENTO: Requisito – RNF001		
*Elemento herdado do projeto **Autenticação de Usuário e Controle de Acesso***		
Identificação	CRM_RNF001	
Customizável	Sim	
Elemento base	AUCA_RNF001	
Elementos customizados	*É Nulo*	
PROPRIEDADES DO ELEMENTO: Requisito – RNF001		
Nome	**Valor**	**Valor herdado**
Tipo	Não Funcional	Sim
Descrição	Apenas usuários devidamente autenticados e do tipo administrador poderão acessar o sistema	Sim

Quadro 36 Elemento: Requisito – RNF002 (CRM)

ELEMENTO: Requisito – RNF002		
Elemento herdado do projeto ***Autenticação de Usuário e Controle de Acesso***		
Identificação	CRM_RNF002	
Customizável	Sim	
Elemento base	AUCA_RNF002	
Elementos customizados	*É Nulo*	
PROPRIEDADES DO ELEMENTO: Requisito – RNF002		
Nome	**Valor**	**Valor herdado**
Tipo	Não Funcional	Sim
Descrição	Apenas os módulos indicados como restritos deverão ter seu acesso mapeado aos usuários. Módulos não restritos podem ser acessados por todos os usuários do sistema	Sim

Quadro 37 Elemento: Requisito – RNF003 (CRM)

ELEMENTO: Requisito – RNF003			***Obs.***
Elemento ***NÃO*** *foi herdado*			***1***
Identificação	CRM_RNF003		
Customizável	Sim		
Elemento base	*É Nulo*		***1***
Elementos customizados	*É Nulo*		
PROPRIEDADES DO ELEMENTO: Requisito – RNF003			
Nome	**Valor**	**Valor herdado**	
Tipo	Não Funcional	Não	
Descrição	O administrador poderá realizar qualquer operação no sistema. Ele terá acesso à todos os módulos	Não	

Observações sobre o Quadro:

(numeração correspondente indicada no quadro, coluna Obs.)

Obs. 1 Este elemento é próprio do projeto de software customizado, não foi herdado de nenhum outro projeto.

6.2.1.3 Análise e Projeto: Modelo de Dados: Tabela

As tabelas presentes no software customizado CRM terão as propriedades definidas para o tipo de elemento “Modelo de Dados: Tabela” (Tabela 4), bem como as propriedades de customização, definidas e descritas no Capítulo 3.

O Quadro 38 ao Quadro 42 mostram os elementos do tipo “Modelo de Dados: Tabela” e o mapeamento das propriedades desses elementos.

Quadro 38 Elemento: Modelo de Dados: Tabela – LOGRADOURO (CRM)

ELEMENTO: Modelo de Dados: Tabela – LOGRADOURO			***Obs.***
*Elemento herdado do projeto “**Cadastro de CEP**”*			
Identificação	CRM_LOGRADOURO		
Customizável	Sim		
Elemento base	CadastroCEP_logradouro		
Elementos customizados	*É Nulo*		
PROPRIEDADES DO ELEMENTO: Modelo de Dados: Tabela – LOGRADOURO			
Nome	**Valor**	**Valor herdado**	
Nome	CEP_LOGRADOURO	Sim	
Colunas	COD_LOGRADOURO;CEP;LOGRADOURO;BAIRRO; COD_ESTADO;COD_MUNICIPIO	Sim	

Quadro 39 Elemento: Modelo de Dados: Tabela – PESSOA (CRM)

ELEMENTO: Modelo de Dados: Tabela – PESSOA			***Obs.***
*Elemento herdado do projeto “**Cadastro de Pessoas**”*			
Identificação	CRM_PESSOA		
Customizável	Sim		
Elemento base	CadastroPessoas_pessoa		
Elementos customizados	*É Nulo*		
PROPRIEDADES DO ELEMENTO: Modelo de Dados: Tabela – PESSOA			
Nome	**Valor**	**Valor herdado**	
Nome	PESSOA	Sim	
Colunas	COD_PESSOA;NOME;CPF;DT_NASC; ENDERECO;TELEFONE;EMAIL;TIPO_PESSOA; NOME_CONTATO;SITE	Não	***1***

Observações sobre o Quadro:

(numeração correspondente indicada no quadro, coluna Obs.)

Obs. 1 O valor da propriedade Colunas foi alterado. Foram incluídas novas colunas na tabela PESSOA. Como este elemento foi herdado de outro projeto, o atributo “Valor Herdado” das propriedades do elemento é definido com valor Não.

Quadro 40 Elemento: Modelo de Dados: Tabela – MODULO (CRM)

ELEMENTO: Modelo de Dados: Tabela – MODULO			***Obs.***
Elemento herdado do projeto ***Autenticação de Usuário e Controle de Acesso***			
Identificação	CRM_MODULO		
Customizável	Não		
Elemento base	AUCA_modulo		
Elementos customizados	*É Nulo*		
PROPRIEDADES DO ELEMENTO: Modelo de Dados: Tabela – MODULO			
Nome	**Valor**	**Valor cerdado**	
Nome	LOGIN_MODULO	Sim	
Colunas	COD_MODULO;MODULO;DESCRICAO; ACESSO_RESTRITO	Sim	

Quadro 41 Elemento: Modelo de Dados: Tabela – ACESSO (CRM)

ELEMENTO: Modelo de Dados: Tabela – ACESSO			***Obs.***
Elemento herdado do projeto ***Autenticação de Usuário e Controle de Acesso***			
Identificação	CRM_ACESSO		
Customizável	Não		
Elemento base	AUCA_acesso		
Elementos customizados	*É Nulo*		
PROPRIEDADES DO ELEMENTO: Modelo de Dados: Tabela – ACESSO			
Nome	**Valor**	**Valor herdado**	
Nome	LOGIN_USUARIO_ACESSO	Sim	
Colunas	COD_USUARIO;COD_MODULO	Sim	

Quadro 42 Elemento: Modelo de Dados: Tabela – CONTATO (CRM)

ELEMENTO: Modelo de Dados: Tabela – CONTATO			***Obs.***
Elemento ***NÃO*** *herdado*			
Identificação	CRM_CONTATO		
Customizável	Sim		
Elemento base	*É Nulo*		
Elementos customizados	*É Nulo*		
PROPRIEDADES DO ELEMENTO: Modelo de Dados: Tabela – CONTATO			
Nome	**Valor**	**Valor herdado**	
Nome	CRM_CONTATO	Não	
Colunas	COD_CONTATO;DATA;COD_USUARIO; COD_TIPO_CONTATO;COD_CLIENTE;CONTATO	Não	

6.2.1.4 Implementação: Componente do Software

Os componentes de software presentes no software customizado CRM terão as propriedades definidas para o tipo de elemento “Componente” (Tabela 4), bem como as propriedades de customização definidas e descritas no Capítulo 3.

O Quadro 43, Quadro 44 e Quadro 45 mostram os elementos do tipo “Componente” e o mapeamento das propriedades desses elementos.

Quadro 43 Elemento: Componente – controleLOGRADOURO (CRM)

ELEMENTO: Componente – controleLOGRADOURO			***Obs.***
*Elemento herdado do projeto “**Cadastro de CEP**”*			
Identificação	CRM_controleLOGRADOURO		
Customizável	Sim		
Elemento base	CadastroCEP_controleLogradouro		
Elementos customizados	*É Nulo*		
PROPRIEDADES DO ELEMENTO: Componente – controleLOGRADOURO			
Nome	**Valor**	**Valor herdado**	
Nome	controleLogradouro.java	Sim	
Corpo	.../*corpo da classe controleLogradouro.java*/...	Não	***1***

Observações sobre o Quadro:

(numeração correspondente indicada no quadro, coluna Obs.)

Obs. 1 O valor da propriedade Corpo foi alterado. Como este elemento foi herdado de outro projeto, o atributo “Valor herdado” das propriedades do elemento é definido com valor Não.

Quadro 44 Elemento: Componente – controlePESSOA (CRM)

ELEMENTO: Componente – controlePESSOA			***Obs.***
*Elemento herdado do projeto “**Cadastro de Pessoas**”*			
Identificação	CRM_controlePESSOA		
Customizável	Sim		
Elemento base	CadastroPessoas_controlePessoa		
Elementos customizados	*É Nulo*		
PROPRIEDADES DO ELEMENTO: Componente – controlePESSOA			
Nome	**Valor**	**Valor Hherdado**	
Nome	controlePessoa.java	Sim	
Corpo	.../*corpo da classe controlePessoa.java*/...	Não	

Quadro 45 Elemento: Componente – controleCONTATO (CRM)

ELEMENTO: Componente – controleCONTATO			***Obs.***
*Elemento **NÃO** herdado*			
Identificação	CRM_controleCONTATO		
Customizável	Sim		
Elemento base	*É Nulo*		***1***
Elementos customizados	*É Nulo*		
PROPRIEDADES DO ELEMENTO: Componente – controleCONTATO			
Nome	**Valor**	**Valor herdado**	
Nome	controleContato.java	Não	
Corpo	.../*corpo da classe controleContato.java*/...	Não	

Observações sobre o Quadro:

(numeração correspondente indicada no quadro, coluna Obs.)

Obs. 1 Como este elemento não foi herdado de nenhum projeto, é próprio do projeto CRM o valor do atributo “Elemento base” será nulo.

6.2.1.5 Testes: Classe de Teste

As classes de teste presentes no software base “Cadastro de CEP” terão as propriedades definidas para o tipo de elemento “Classe de Teste” (Tabela 4), bem como as propriedades de customização, definidas e descritas no Capítulo 3.

O Quadro 46 e Quadro 47 mostram os elementos do tipo “Classe de Teste” e o mapeamento das propriedades desses elementos.

Quadro 46 Elemento: Classe de Teste – tControleLOGRADOURO (CRM)

ELEMENTO: Classe de Teste – tControleLOGRADOURO			***Obs.***
*Elemento herdado do projeto **Cadastro de CEP***			
Identificação	CRM_tControleLOGRADOURO		
Customizável	Sim		
Elemento base	CadastroCEP_tControleLogradouro		
Elementos customizados	*É Nulo*		
PROPRIEDADES DO ELEMENTO: Classe de Teste – tControleLOGRADOURO			
Nome	**Valor**	**Valor herdado**	
Nome	tControleLogradouro.java	Sim	
Operações	tInsertLogradouro (Logradouro newLogradouro);tDeleteLogradouro (int codLogradouro);tUpdateLogradouro (Logradouro newLogradouro, Logradouro oldLogradouro);	Sim	

Quadro 47 Elemento: Classe de Teste – tControleLOGIN (CRM)

ELEMENTO: Classe de Teste – tControleLOGIN			***Obs.***
Elemento herdado do projeto ***Autenticação de Usuário e Controle de Acesso***			
Identificação	CRM_tControleLOGIN		
Customizável	Sim		
Elemento base	AUCA_tControleLOGIN		
Elementos customizados	*É Nulo*		
PROPRIEDADES DO ELEMENTO: Classe de Teste – tControleLOGIN			
Nome	**Valor**	**Valor cerdado**	
Nome	tControleLogin.java	Sim	
Operações	tLoginUsuario (String usuario, String senha); tTrocaSenha (String usuario, String senhaAtual, String novaSenha)	Sim	

7 ANEXO II - ARTIGO

Proposta para o Gerenciamento de Customizações de Software em seu processo de desenvolvimento

Alexsandra Duarte Borges

Universidade Federal de Santa Catarina (UFSC) – Depto. de Informática e Estatística – Florianópolis, SC

aledb@inf.ufsc.br

Abstract. *This study presents a proposal for software customizations management during all software development process. The proposal follows a software customization model by levels, which allows creating a structure where the software is developed from other ready software projects.*

Resumo. *Este trabalho apresenta uma proposta para o gerenciamento de customizações de software realizadas durante todo o processo de desenvolvimento do software. A proposta segue um modelo de customização de softwares em níveis, que permite a criação de uma estrutura onde softwares são desenvolvidos a partir de outros projetos de softwares.*

1. Introdução

A rapidez com que circulam as informações, a disponibilidade de ferramentas e o surgimento contínuo de novas tecnologias permitiram o estabelecimento de um ambiente altamente competitivo para o mercado de software em que variáveis como: bom atendimento, custo, tempo e flexibilidade sejam o diferencial para que as empresas consigam conquistar novos clientes e manter os antigos fidelizados.

Segundo Roselino (2006, p. 29) há um grande esforço por parte dos engenheiros de software na procura de modelos maduros de desenvolvimento que permitam a racionalização em busca de ganhos de produtividade no processo de produção. Mesmo quando uma empresa desenvolvedora de software já possua a solução (pacote de aplicativos) a ser adquirido pelo cliente, o que pode tornar o processo de implantação naturalmente mais rápido, cada um destes clientes possui características e necessidades específicas. Neste caso, ou a empresa limita a personalização das funcionalidades do produto existente, reduzindo o custo de mão-de-obra e conseqüentemente seu valor de venda ou a empresa eleva o valor do produto e gasta um tempo considerável para modificar a ferramenta e adequá-la as necessidades do cliente.

Tendo em vista o alto custo de operacionalização deste processo, a criação de um modelo que permita um adequado gerenciamento de customizações de software trará à equipe de

desenvolvimento flexibilidade na geração de novas funcionalidades e na remodelagem de funcionalidades já existentes. Com isso, o ganho de produção, maleabilidade e tempo beneficiaria ambas as partes, tanto o fornecedor do produto, quanto do cliente que o adquire.

2. Customização de Software

Segundo Piacentini (2008) customização "tem o sentido de adaptar os produtos e processos ao gosto do cliente, portanto é o atendimento que visa a satisfação do freguês". Sua origem também pode ser associada ao termo em inglês "*custom*" que quer dizer feito sob medida ou personalizado. Desta forma, customizar um software é redesenhá-lo ou adequá-lo às necessidades de quem o irá usar. No desenvolvimento essa customização pode ser realizada de duas formas distintas: já no processo de concepção e análise (na definição do software) ou com o produto já pronto, quando apenas algumas características e funcionalidades são modificadas para que este se molde aos novos requisitos.

Na abordagem em que a customização é realizada durante o processo de desenvolvimento, um projeto de software acabado (pronto) pode servir como base para a criação de um novo software. Este voltado a um cliente específico que demandará as necessidades não atendidas pelo software base. Ou seja, o software se adaptará ao cliente através da sua customização.

2.1 Um Modelo de Customização de Software

O objetivo de se fazer customizações de software, segundo a abordagem deste trabalho, vem do princípio de se utilizar um projeto de software pronto que atenda todas ou parte das necessidades do cliente e apenas incrementá-lo ou modificá-lo quando necessário, como mostra a Figura 1. Neste caso, cada versão customizada possui todas as características do software base e mais as características e requisitos que não são contemplados por este, e que são criados e modelados especificamente em cada versão.

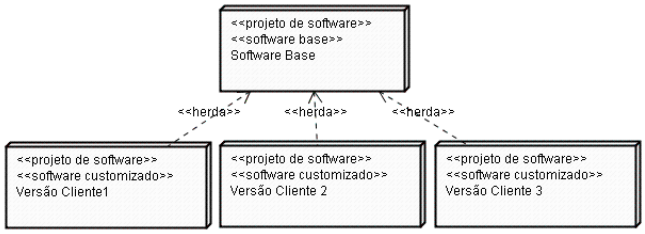

Figura 1 Softwares customizados para dois clientes distintos que utilizam dois e um softwares base

Partindo do princípio de que cada projeto customizado é criado a partir de projetos de software base, e que estes projetos tem uma relação de dependência com os utilizados para compô-lo, pode-se atingir o estágio em que os próprios projetos customizados podem servir

de base para outros projetos. A Figura 2 representa uma estrutura em que projetos de software base compõem projetos customizados que se transformam em base para novos projetos.

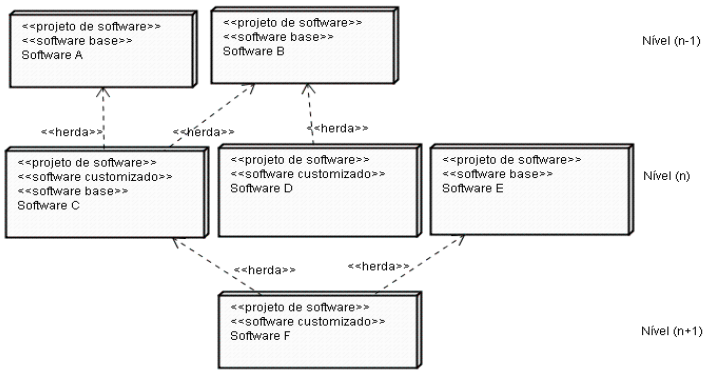

Figura 2 Diagrama de Utilização de customizações de projetos de software em níveis

A customização de software, conforme o modelo de customização descrito pode gerar uma árvore com diferentes níveis de hierarquia, em que um software P(n) herda características de outro software P(n-1). Ou seja, todos os artefatos produzidos no projeto P(n-1) são herdados pelo projeto P(n). A figura 3 mostra como seria o diagrama de classes de projetos de software com customização, bem como a relação destes projetos com seus elementos[7].

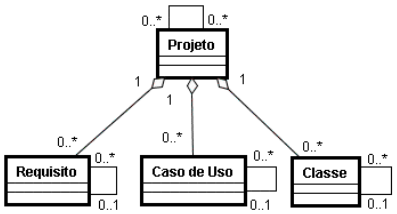

Figura 3 Diagrama de Classes Projetos com Customização

2. Gerenciamento da Customização de Software

De uma forma prática, ao se customizar um software, na verdade, estão sendo realizadas customizações nas propriedades dos elementos que compõem esse software, ou seja, nos atributos das classes dos diferentes tipos de elementos. A proposta deste trabalho é baseada na geração de um modelo de classes para o gerenciamento das customizações realizadas no software. Este modelo poderá ser estendido por projetos em que se deseje realizar o gerenciamento de customizações de software. Esse gerenciamento poderá ser aplicado e

[7] O termo elemento é utilizado para definir qualquer artefato de software produzido durante o processo de desenvolvimento do software.

mapeado nos três diferentes níveis que compõem um projeto de software: Projeto, Elemento e Propriedade.

O gerenciamento da customização de projetos trata da relação existente entre projetos base e projetos customizados. Isso será administrado por uma classe de gerenciamento, denominada CProjeto, que possui informações referentes à customização e à extensibilidade do projeto de software. Esta classe deverá ser estendida pela classe Projeto para que seja possível efetuar o gerenciamento de customizações dos projetos e, desta forma, além dos atributos inerentes a um projeto de software, o projeto terá atributos referentes à sua customização.

O gerenciamento de customizações do elemento de software trata a relação existente entre um elemento herdado e seu elemento base. Isso será administrado por uma classe de gerenciamento, denominada CElemento, que possui informações referentes a customizações do elemento de software. Esta classe deverá ser estendida por todas as classes de elemento de software para que seja possível efetuar o gerenciamento de customizações dos elementos e, desta forma, além dos atributos inerentes a um tipo específico de elemento, o elemento terá atributos referentes à sua customização.

O gerenciamento das customizações das propriedades dos elementos trata a conformidade do valor da propriedade do elemento herdado em relação ao valor da propriedade correspondente no elemento base. Isso será administrado por uma classe de gerenciamento, denominada CPropriedade. Esta classe está relacionada à classe CElemento e representará cada uma das propriedades de um tipo específico de elemento.

Desta forma, o diagrama de classes que representa os projetos de software com customização estenderá as classes descritas para que seja efetuado o gerenciamento de customizações de software, conforme figura 4.

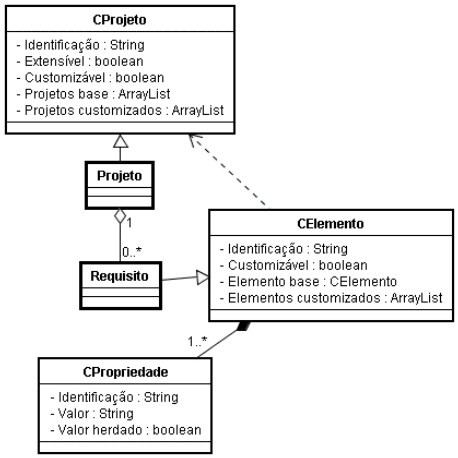

Figura 4 Diagrama de Classes de Projetos com Gerenciamento de Customização

2.1. Restrições para o Gerenciamento de Customizações de Software

Como são muitos os elementos de artefatos de software a serem gerenciados em um projeto de software e as propriedades, tanto as de customização quanto as inerentes a cada tipo de elemento podem ou não ser modificadas de um projeto para outro, é importante que se crie um procedimento para modificação dos valores dessas propriedades. Por este motivo a operacionalização do gerenciamento de customizações de software deve atender a regras relacionadas aos três níveis que compõem um projeto (Projeto, Elemento e Propriedade) e que permitirão a manutenção da integridade do modelo. Estas regras poderão ser incorporadas ao metamodelo de gerenciamento através de métodos implementados nas classes de gerenciamento da customização.

2. Aplicação da Proposta de Gerenciamento de Customizações em Projetos de Software Exemplo

Para demonstrar a prática do gerenciamento de customizações de software: seu funcionamento, a definição das propriedades de customização e observância das regras de gerenciamento, foi utilizado um conjunto de elementos produzidos em algumas das etapas do desenvolvimento de um software, bem como alguns projetos de software base. Os projetos de software utilizados como base para aplicação são: "Cadastro de CEP", "Cadastro de Pessoas" e "Autenticação de Usuários e Controle de Acesso". O software customizado deverá operar a administração do cadastro dos clientes de uma operadora de *telemarketing*, bem como controlar os contatos efetuados com eles. A figura 5 apresenta o detalhamento de um elemento do projeto customizado que foi herdado do projeto de software base "Cadastro de CEP". A definição dos elementos e propriedades de elementos foi orientada pela metodologia de desenvolvimento de software RUP (*Rational Unified Process*).

Quadro 1 Elemento: Caso de Uso – UC001 (CRM)

ELEMENTO: Caso de Uso – UC001		
*Elemento **herdado** do projeto "**Cadastro de CEP**"*		
Identificação	CRM_UC001	
Customizável	Sim	
Elemento base	CadastroCEP_UC001	
Elementos customizados	*É Nulo*	
PROPRIEDADES DO ELEMENTO: Caso de Uso – UC001		
Nome	**Valor**	**Valor Herdado**
Nome	Cadastrar CEP	Sim
Breve Descrição	O Operador poderá cadastrar, alterar e excluir registros de CEP's.	Sim

2. Conclusões

Pensar na customização de um software desde a sua concepção até sua implantação e manutenção pode trazer inúmeros benefícios. Dentre eles, a facilidade em gerenciar requisitos de um produto implantado em vários clientes ou mesmo a adaptação da interface a um usuário específico ou a perfis de usuários em uma mesma versão da aplicação. Com isso, modificações simples no processo de desenvolvimento do software e a adoção de um modelo de gerenciamento favorecem tanto o fornecedor, desenvolvedor do software, que poderá agilizar implantações e manutenções de seu produto quanto o cliente que terá uma resposta rápida ao atendimento de suas necessidades. Mesmo com a existência de dificultadores, inerentes a aplicação de uma nova forma de se desenvolver software, a customização e o gerenciamento de customizações de software têm o potencial para ser uma importante contribuição à relação qualidade, tempo e custo, tão discutida atualmente pelos que fornecem e pelos que adquirem software.

Referências

KRUCHTEN, Philippe. **Introdução ao RUP – *Rational Unified Process***. 2ª Edição, Rio de Janeiro: Editora Ciência Moderna Ltda., 2003. 255p.

PIACENTINI, Maria Tereza de Queiroz. **Customizar, fulcrar, ociar**. Disponível em:<http://kplus.cosmo.com.br/materia.asp?co=148&rv=Gramatica>. Acesso em: 14 jun. 2008

RATIONAL Software Corporation. **Rational Unified Process**. Versão 2002.05.00 Rational, 2002. CD-ROM.

ROSELINO, José Eduardo. **A INDÚSTRIA DE SOFTWARE: o "modelo brasileiro" em perspectiva comparada**. 2006. 236 f. Tese (Doutorado) - Curso de Ciências Econômicas, Departamento de Instituto de Economia, Universidade Estadual de Campinas, São Paulo, 2006.

SILVA, Ricardo Pereira. **UML 2:** Modelagem Orientada a Objetos. Florianópolis: Visual Books, 2007. 232 p.