UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

JAYLON HENRIQUE DA SILVA

IMPLEMENTAÇÃO DE UM SISTEMA WEB PARA GESTÃO DE CONFIGURAÇÃO EM PROJETOS DE SOFTWARE

TRABALHO DE CONCLUSÃO DE CURSO

PATO BRANCO
2014

JAYLON HENRIQUE DA SILVA

IMPLEMENTAÇÃO DE UM SISTEMA WEB PARA GESTÃO DE CONFIGURAÇÃO EM PROJETOS DE SOFTWARE

Trabalho de Conclusão de Curso de graduação, apresentado à disciplina de Trabalho de Diplomação, do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, da Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco, como requisito parcial para obtenção do título de Tecnólogo.

Orientador: Profa. Eliane De Bortoli Fávero

PATO BRANCO
2014

ATA Nº: **256**

DEFESA PÚBLICA DO TRABALHO DE DIPLOMAÇÃO DO ALUNO **JAYLON HENRIQUE DA SILVA.**

Às 17:30 hrs do dia 18 de dezembro de 2014, Bloco V da UTFPR, Câmpus Pato Branco, reuniu-se a banca avaliadora composta pelos professores Eliane Maria de Bortoli Fávero (Orientadora), Beatriz Terezinha Borsoi (Convidada) e Alessandro Graczyk Moraes (Convidado), para avaliar o Trabalho de Diplomação do aluno Jaylon Henrique Da Silva, matrícula 1375504, sob o título **Implementação de um Sistema Web para Gestão de Configuração em Projetos de Software**; como requisito final para a conclusão da disciplina Trabalho de Diplomação do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, COADS. Após a apresentação o candidato foi entrevistado pela banca examinadora, e a palavra foi aberta ao público. Em seguida, a banca reuniu-se para deliberar considerando o trabalho **APROVADO**. Às 18:05 hrs foi encerrada a sessão.

Profa. Eliane Maria de Bortoli Fávero, M.Sc.
Orientadora

Profa. Beatriz Terezinha Borsoi, Dr.
Convidada

Prof. Alessandro Graczyk Moraes,
Convidado

Profa. Eliane Maria de Bortoli Fávero, M.Sc
Coordenadora do Trabalho de Diplomação

Prof. Edilson Pontarolo, Dr.
Coordenador do Curso

RESUMO

SILVA, Jaylon Henrique. Implementação de um Sistema Web para Gestão de Configuração em Projetos de Software. 2014. 60f. Monografia (Trabalho de Conclusão de Curso) - Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas. Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco. Pato Branco, 2014.

Este trabalho aborda uma atividade relativamente crítica no desenvolvimento de um produto de software: a gestão de configuração. Sua importância está inerente ao processo de desenvolvimento de software, ao longo do qual surgem necessidades de mudança nos artefatos gerados, sendo necessário registrá-las, e controlá-las visando manter a integridade do projeto durante sua evolução. Sendo assim, esse trabalho visa apresentar uma ferramenta para auxílio às atividades relativas à gestão de configuração, implementada como complemento ao software de gestão de projetos utilizado pela disciplina de Oficina de Projeto e Desenvolvimento de Software do Curso de Tecnologia em Análise e Desenvolvimento de Sistemas, da Universidade Tecnológica Federal do Paraná para simular uma fábrica de software. O sistema de software desenvolvido abrange os processos inerentes à gestão de configuração, definidos no nível F do modelo de qualidade Melhoria do Processo de Software Brasileiro (MPS.BR). Desta forma, o sistema implementa a definição e gerenciamento de *baselines*, que são instâncias compostas por todos os artefatos relacionados ao processo de desenvolvimento de um projeto de software, em um período determinado. Os artefatos que compõe estas *baselines* são denominados Itens de Configuração (IC), os quais podem sofrer mudanças durante o período de desenvolvimento do projeto. Estas mudanças são registradas pelo sistema, a partir da aprovação das respectivas solicitações de mudanças, após a análise de impacto dessas alterações. Com base no nível de impacto da alteração realizada sobre um IC, é permitida a mudança de versão do IC, sendo permitido que os usuários possam recuperar versões específicas de um IC. Embora a solução seja projetada para ser complemento ao sistema de gestão de projetos da disciplina de Oficina de Projeto e Desenvolvimento de Software, ela é aplicável à ambientes reais de desenvolvimento de fábricas de software de qualquer porte. Esse sistema foi desenvolvido para a plataforma *Web*, destacando-se dentre as tecnologias utilizadas a linguagem Java, com os *frameworks JavaServer Faces*, acompanhado da biblioteca *PrimeFaces* para desenvolvimento das interfaces e Hibernate para persistência de dados.

Palavras-chave: Gestão de configuração. Java para *Web*. *JavaServer Faces*. *PrimeFaces*. MPS.BR.

ABSTRACT

SILVA, Jaylon Henrique. Implementation of a Web System to Configuration Management in Software Projects. 2014. 60f. Monografia (Trabalho de Conclusão de Curso) - Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas. Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco. Pato Branco, 2014.

.

This work addresses a relatively critical activity in the software product development: configuration management. His importance is inherent to the software development process, along which arise change necessities in the generated artifacts, being necessary to register them, and control them to keep the project integrity during his evolution. Therefore, this work aims to present one tool to assist the relative activities to configuration management, implemented as complement to the projects management software used by the discipline of Design Project and Software Development from course of Technology in Analysis and Systems Development, of Federal Technological University of Paraná to simulate a software factory. The software system developed covers the processes inherent to the configuration management, defined in the level F from the quality model called Process Improvement of Brazilian Software (MPS.BR). In this way, the system implements the definition and management of baselines, who are instances composite for all the related artifacts to the one project software development process, in a given period. The artefacts that compose these baselines are called Configuration Items (CI), which can undergo changes during the project development period. These changes are registered by the system, from the approval of respective change requests, after the impact analysis of these changes. With base in the impact level of a change made in a CI, it is allowed to change the CI version, allowing the users to retrieve specific versions of one CI. Although the solution has been developed to be a complement to an existing system, it is applicable to real environments, as software factories with any size. This system was developed to the web platform, standing out from the used technologies the Java language, with JavaServer Faces framework, accompanied of *PrimeFaces* library to develop the interfaces and Hibernate to data persistence.

Key-words: Configuration Management, Java for *Web*. *JavaServer Faces*. *PrimeFaces*. MPS.BR.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE LISTAGENS

LISTA DE SIGLAS

BID	Banco Interamericano de Desenvolvimento
CMM	*Capability Maturity Model*
CMMI	*Capability Maturity Model Integration*
CRUD	*Create, Read, Update e Delete*
CSS	*Cascade Style Sheet*
DAO	*Data Access Objects*
FINEP	Financiadora de Estudos e Projetos
GCO	Gerência de Configuração
GCS	Gestão de Configuração de Software
GDE	Gerência de Decisões
GPR	Gerência de Projetos
GQS	Garantia de Qualidade de Software
GRE	Gerência de Requisitos
HTML	*Hypertext Markup Language*
IC	Item de Configuração
IDE	*Integrated Development Environment*
ITP	Integração do Produto
JPA	*Java Persistence API*
JSF	*JavaServer Faces*
JTA	*Java Transaction API*
MCT	Ministério da Ciência e Tecnologia
MPS.BR	Melhoria de Processos do Software Brasileiro
MR-MPS	Modelo de Referência de Melhoria de Processos de Software
SEBRAE	Serviço Brasileiro de Apoio às Micro e Pequenas Empresas
SOFTEX	Associação para Promoção da Excelência do Software Brasileiro
SVN	*Subversion*
SQL	*Structured Query Language*
URL	*Uniform Resource Locator*
XHTML	*eXtensible Hypertext Markup Language*

SUMÁRIO

1 INTRODUÇÃO

Esse capítulo apresenta as considerações iniciais para o trabalho, inserindo-o no contexto do desenvolvimento de software e definindo o trabalho a ser realizado.

1.1 CONSIDERAÇÕES INICIAIS

Durante o ciclo de vida de um projeto de software muitas mudanças acontecem, sendo estas inerentes às tecnologias, questões organizacionais, requisitos, dentre outros motivos. A fim de manter a integridade do projeto, é de suma importância a existência de técnicas e ferramentas para o gerenciamento eficaz das mudanças ocorridas durante a execução de um projeto.

Segundo MACHADO (2001), para muitos engenheiros de software, a qualidade do processo de software é tão importante quanto a qualidade do produto. Portanto, estas mudanças estão fortemente relacionadas à qualidade, pois essa depende diretamente da manutenção da integridade do projeto. Para atender esta premissa, foram estipulados padrões de Garantia de Qualidade de Software (GQS), que segundo o IEEE-STD-610 (1990) representam um padrão planejado e sistemático de ações para prover uma confiança adequada de que um item está em conformidade com os requisitos técnicos estabelecidos.

Com base nas atividades relacionadas à GQS surgiram modelos de qualidade de software nos quais, de alguma forma, a gestão de mudanças é implementada. O modelo de Melhoria do Processo de Software Brasileiro (MPS.BR) define uma área específica para controle de mudanças e processos inerentes à elas, que segundo o SOFTEX (2011) denomina-se Gestão de Configuração de Software (GCS).

A implantação da GCS, que pode ser abordada de forma diferente por cada modelo de qualidade, mesmo que não implantada de forma integral, pode representar um diferencial para empresas, independente de recursos financeiros e humanos.

Este trabalho aborda uma tarefa relativamente crítica no desenvolvimento de um produto de software: a gestão de configuração. Sua importância se dá em função da grande demanda de alterações em artefatos de um projeto durante o seu desenvolvimento, pois, à medida que surgem as necessidades de mudança é necessário absorvê-las e controlá-las para que seja mantida a estabilidade e integridade na evolução do projeto.

Por isso, este trabalho propõe o desenvolvimento de um aplicativo na plataforma Web que permita a realização das práticas de gestão de configuração definidas no nível F do MPS.BR. O sistema proposto abrangerá desde a definição de itens de configuração, a manutenção e liberação de *baselines*, a gestão de mudança sobre os itens de configuração que as compõe, bem como a manutenção das versões de cada item de configuração. O produto final deste trabalho será adicionado como complemento ao software de Gestão de Projetos que será utilizado pela disciplina de Oficina de Projeto e Desenvolvimento de Software.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

- Implementar um sistema Web para gestão de configuração no processo de desenvolvimento de software, o qual será usado na disciplina de Projeto e Desenvolvimento de Software.

1.2.2 Objetivos Específicos

- Permitir a gestão de itens de configuração de um projeto de software em termos de modificações, análise de impacto, versionamento e liberação de itens de configuração.
- Contribuir para a manutenção da integridade de projetos de software considerando a sua dinâmica de mudanças.

- Facilitar o trabalho da gerência de um projeto de software, bem como da equipe de projeto com relação às modificações, versionamento e liberações de itens de configuração.

1.3 JUSTIFICATIVA

O projeto será desenvolvido visando sanar as necessidades demandadas por empresas produtoras de software em relação à gestão de configuração, adaptando a implementação das funcionalidades à utilização na disciplina de Oficina de Projeto e Desenvolvimento de Software, onde a ferramenta servirá de apoio às tarefas inerentes à gestão de projetos, abrangendo a gestão, manutenção e liberação de *baselines*, bem como a manutenção e controle de mudanças sobre os itens de configuração que as compõem.

Apesar da criticidade das práticas da gestão de configuração, não existem muitas ferramentas gratuitas que permitam executá-las. Em virtude disto, muitos projetistas utilizam-se de ferramentas de versionamento para executar esta tarefa. Embora seja possível executar boa parte das práticas de gestão de configuração com estas ferramentas, algumas delas, como a gestão de mudanças e configuração de *baselines* são atendidas parcialmente ou sequer são atendidas.

Em relação às tecnologias, a sua escolha se deu devido ao projeto existente da disciplina de Projeto e Desenvolvimento de Software, onde este projeto será acoplado. O projeto será desenvolvido para a plataforma Web (modelo cliente/servidor), utilizando da tecnologia JSF (*Java Server Faces*) implementada pelo framework *PrimeFaces*, onde a estrutura do projeto é organizada segundo o modelo MVC (*Model View Control*), separando assim as camadas de negócio, controle e visão (interação com o usuário).

1.4 ESTRUTURA DO TRABALHO

O Capítulo 1 representa uma contextualização do trabalho, seus objetivos e justificativa. No Capítulo 2 é apresentado um levantamento teórico sobre aspectos conceituais da GCS sobre a perspectiva geral e de desenvolvimento, analisando sua aplicação nos modelos de qualidade CMM, CMMI e MPS.BR. Por fim, aborda quatro ferramentas que permitem a implementação da GCS, dentre estas as ferramentas JIRA e CONFLUENCE abordam a GCS no âmbito de gestão de projetos, e as ferramentas SUBVERSION e GIT que implementam a GCS por versionamento de arquivos. Já no Capítulo 3 são apresentados os materiais e o método utilizados na elaboração do projeto. O Capítulo 4 apresenta o escopo, requisitos e a modelagem para o sistema, finalizando com a apresentação do sistema e de sua implementação. No Capítulo 5 são apresentadas as conclusões do trabalho.

2 REFERENCIAL TEÓRICO

A seguir serão apresentados os conceitos e referenciais envolvidos na Gestão de Configuração de Software, bem como a análise de ferramentas destinadas à GCS, as quais foram julgadas como necessárias para o desenvolvimento do trabalho proposto.

2.1 CONCEITO E CONTEXTO DA GCS

Ao longo do ciclo de vida de um projeto de software, uma grande quantidade de itens de informação é produzida. Como exemplos desses itens são possíveis citar documentos, código fonte, dados, manuais, etc. É natural que muitos desses itens passem por modificações no decorrer do projeto em função de diversos motivos, tais como mudança nos requisitos ou correção de defeitos. Isto traz como consequência o aumento da complexidade no desenvolvimento e gerenciamento de software durante seu processo de criação e/ou manutenção, assim como possíveis inconsistências entre os itens gerados nesses processos.

Em ambientes de desenvolvimento concorrente esta situação é agravada, pois duas ou mais pessoas podem estar trabalhando no mesmo artefato, tornando-se indispensável um controle de acesso e de versões de cada artefato, buscando minimizar as inconsistências. Portanto, é necessário gerenciar os sistemas em desenvolvimento porque, à medida que eles são alterados, são criadas muitas versões diferentes do software (SOMMERVILLE, 2003, p. 550). Complementando as palavras do Sommerville, são criadas muitas versões de artefatos relacionados ao sistema de software (ex. documentos, diagramas, casos de teste, código-fonte). Apesar dessa necessidade tão clara, muitas empresas não se preocupam com estes aspectos e acabam comprometendo seus orçamentos e cronogramas, e principalmente, fazendo com que a qualidade de seu serviço seja baixa.

A fim de amenizar esses problemas, na grande área denominada Engenharia de Software existe a Gerência de Configuração de Software, na qual se definem

etapas para os trabalhos em equipe e controle dos artefatos no ciclo de vida do produto (SOMMERVILLE, 2003, p. 550).

> A GCS é uma disciplina que aplica procedimentos técnicos e administrativos para identificar e documentar as características físicas e funcionais de um Item de Configuração[1] (IC) controlar as alterações nessas características, armazenar e relatar o processamento das modificações, o estágio da implementação e verificar a compatibilidade com os requisitos especificados (IEEE, 1990).

Desta maneira, a GCS não se propõe a definir quando e como devem ser executadas as modificações nos artefatos de software, papel este reservado ao próprio processo de desenvolvimento de software. A sua atuação ocorre como processo auxiliar de controle e acompanhamento do processo, tendo por objetivo o controle das mudanças ocorridas nos artefatos gerados pelo processo, de forma a garantir a consistência do projeto de software e, com isso, garantir um produto final que satisfaça às expectativas do cliente.

Para ajudar a controlar as mudanças, existem as linhas básicas (*baselines*), que segundo Pressman (1995, p. 918) é um conceito de gerenciamento de configuração de software que nos ajuda a controlar as mudanças, sem impedir seriamente as mudanças justificáveis. Uma linha básica pode ser composta de um ou mais IC. De forma geral, um IC pode ser alterado inúmeras vezes no decorrer do ciclo de desenvolvimento, porém, antes de um IC tornar-se uma linha básica, ele precisa ser aprovado por meio de uma revisão técnica formal, que é uma atividade de garantia de qualidade de software executada por profissionais da engenharia de software (PRESSMAN, 1995, p. 740).

2.2 PADRÕES DE QUALIDADE E A GCS

A GCS, por ser uma área fortemente calcada em controle, inevitavelmente gera uma vasta gama de processos, normas, procedimentos, políticas e padrões.

[1] Item de Configuração é qualquer documento ou grupos de documentos relacionados em controle da configuração do software, como por exemplo: compiladores, códigos-fonte, documentos de análise, *checklists*, etc.

Desta forma, fazendo parte de padrões ou modelos de qualidade de software, ou seja, como os produtos e serviços são planejados, produzidos e entregues. A seguir, será apresentado o modelo de qualidade de software intitulado Melhoria de Processos do Software Brasileiro (MPS.BR) e sua relação com a GCS.

2.2.2 MPS.BR

Segundo o SOFTEX (2011) o MPS.BR é um programa mobilizador, de longo prazo, criado em dezembro de 2003, coordenado pela Associação para Promoção da Excelência do Software Brasileiro (SOFTEX), que conta com apoio do Ministério da Ciência e Tecnologia (MCT), Financiadora de Estudos e Projetos (FINEP), Serviço Brasileiro de Apoio às Micro e Pequenas Empresas (SEBRAE) e Banco Interamericano de Desenvolvimento (BID).

O padrão de processo MPS.BR é simultaneamente um movimento para a melhoria da qualidade e um modelo de qualidade de processo, por isso, também chamado de Modelo de Referência MPS (MR-MPS). Este modelo visa definir e aprimorar um modelo de melhoria e avaliação de processo de software, visando preferencialmente as micro, pequenas e médias empresas, de forma a atender as suas necessidades de negócio e ser reconhecido nacional e internacionalmente como um modelo aplicável à indústria de software. O MPS.BR também estabelece um processo e um método de avaliação, o que dá sustentação e garante que o MPS.BR está sendo empregado de forma coerente com as suas definições (SOFTEX, 2011).

Este modelo define níveis de maturidade que são uma combinação entre processos e sua capacidade. Permitindo avaliar e atribuir graus de efetividade dos processos. Os níveis de maturidade permitem acompanhar a evolução e desempenho ao executar um processo. Segundo SOFTEX (2011), o modelo é dividido em sete níveis de maturidade:

A. **Em Otimização:** é composto pelos processos dos níveis B ao G, incluindo o processo de Implantação de Inovações na Organização e o processo de

Resolução de Causas. O processo de Implantação de Inovações na Organização tem por objetivo implantar melhorias incrementais e inovadoras que contribuam para a melhoria do processo e tecnologia. O processo de Resolução de Causas tem por objetivo identificar a causa de defeitos e outros problemas e fazer a prevenção dos mesmos.

B. **Gerenciado Quantitativamente:** é composto pelos processos dos níveis C ao G, acrescido do processo de Desempenho do Processo Organizacional e Gerência Quantitativa do Projeto. O processo de Desempenho do Processo tem por finalidade estabelecer e manter um entendimento quantitativo do desempenho dos processos-padrão da organização, assim como, fornecer dados, modelos e *baselines* para gerenciar quantitativamente os projetos. O processo de Gerência Quantitativa do Projeto tem por propósito gerenciar quantitativamente o processo definido para o projeto para maximizar a usabilidade do mesmo.

C. **Definido:** é composto pelos processos dos níveis D ao G, acrescido do processo de Análise de Decisão e Resolução e do processo de Gerência de Riscos. O processo de Análise de Decisão e Resolução tem por objetivo analisar possíveis decisões usando um processo formal de avaliação das alternativas identificadas em relação a critérios estabelecidos. O processo de Gerência de Riscos tem por objetivo identificar, gerenciar e minimizar riscos em um âmbito organizacional e de projeto.

D. **Largamente Definido**: é composto pelos processos dos níveis E ao G, incluindo os processos de Desenvolvimento de Requisitos, Integração do Produto, Solução Técnica, Validação e Verificação. No processo de Desenvolvimento de Requisitos devem ser estabelecidos os requisitos dos componentes do produto e do cliente. Já no processo de Integração do Produto devem ser compostos os componentes do produto de forma que eles fiquem consistentes com o projeto e fazer a verificação de adequação dos requisitos funcionais e não-funcionais no ambiente de alvo. No processo de Solução Técnica devem ser projetadas as soluções para atender aos requisitos. O processo de Validação tem por propósito certificar-se de que um produto e/ou componente atenderá o seu uso pretendido no ambiente alvo.

No processo de Verificação o objetivo é confirmar que cada produto e/ou componente reflete apropriadamente os requisitos especificados.

E. **Parcialmente Definido:** é composto pelos processos dos níveis F e G, incluindo os processos de Adaptação do Processo para Gerência do Projeto, Avaliação e Melhoria do Processo Organizacional, Definição do Processo e Treinamento. O processo de Adaptação do Processo para Gerência do Projeto deve estabelecer e gerenciar o projeto envolvendo os interessados. O processo de Avaliação e Melhoria do Processo Organizacional deve consistir na identificação e melhoria contínua da contribuição dos processos padrões da organização para a organização. O processo de Definição do Processo deve estabelecer e manter processos aplicáveis às necessidades de negócio da organização. O processo de Treinamento tem por propósito prover os conhecimentos e habilidades necessárias aos profissionais para determinadas atividades.

F. **Gerenciado:** é composto pelos processos do nível G, incluindo o processo de Aquisição no qual o produto final deve satisfazer a necessidade expressa pelo cliente. Inclui também o processo de Garantia da qualidade que deve garantir que os produtos de trabalho e execução dos processos estão em conformidade com os recursos e cronogramas estabelecidos. Deve também atender ao processo de Medição coletando os dados sobre relativos à produção do produto a fim de apoiar os objetivos organizacionais. Deve abranger a Gerência de configuração mantendo a integridade de todos os produtos do trabalho e disponibilizá-los a todos os envolvidos.

G. **Parcialmente Gerenciado:** não é necessário que se tenha padrões organizacionais. Precisa implementar a gerência de projeto, identificando, estabelecendo, coordenando e monitorando as atividades, tarefas e recursos utilizados na produção do produto. É preciso também implementar a gerência de requisitos que permite identificar inconsistências nos requisitos do produto e componentes do produto.

Esta divisão em um maior número de níveis permite uma avaliação dos resultados obtidos em prazos mais curtos. Têm por objetivo permitir uma implementação adequada às micros, pequenas e médias empresas. A escala de maturidade inicia no nível G e vai até o nível A.

A GCS é abrangida no nível F, sendo assim ela é um dos requisitos para atingir quaisquer outros níveis após o G. Segundo SOFTEX (2011), o escopo do processo de GCS não se aplica unicamente aos produtos de trabalho dos processos de um determinado nível do MPS.BR. Todos os produtos de trabalho dos processos de software em uso pela organização - sejam eles de desenvolvimento, manutenção ou apoio - são considerados. É importante notar que a Gerência de Configuração é um importante mecanismo para aumentar o controle sobre os produtos de trabalho.

O Guia de Implementação do MPS.BR, conforme SOFTEX (2011), estabelece que os níveis de controle podem variar de acordo com a importância ou criticidade dos produtos de trabalho, mas devem ser adequados a cada caso específico. Assim, para produtos de trabalho que requerem um controle mais formal, a Gerência de Configuração é aplicável, tanto no contexto de projetos como no contexto organizacional. Alguns documentos, no entanto, podem ser armazenados com um simples controle de acesso ou, então, serem versionados sem necessidade de um controle formal de mudança. Definir quais produtos de trabalho serão sujeitos a quais níveis de controle é parte da execução do processo de Gerência de Configuração.

O processo Gerência de Configuração está intimamente relacionado com outros processos do MR-MPS. Por exemplo: o processo Gerência de Projetos (GPR) pode apoiar no planejamento do processo Gerência de Configuração; o processo de Gerência de Decisões (GDE) pode apoiar na atividade de avaliação de solicitações de modificação do processo Gerência de Configuração. O processo Gerência de Configuração pode também apoiar o processo Gerência de Requisitos (GRE) no que diz respeito ao controle de modificações sobre os requisitos e o processo Integração do Produto (ITP) no que diz respeito ao controle da evolução de interfaces (SOFTEX, 2011).

A Figura 2 apresenta o relacionamento entre os níveis do modelo CMMI e do MPS.BR.

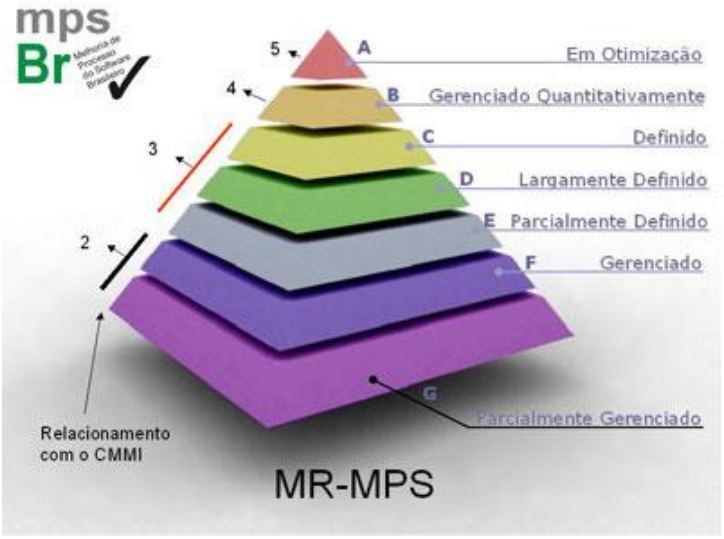

Figura 1 - Relação dos níveis do MPS.BR com os níveis do modelo CMMI
Fonte: SOFTEX (2011).

2.2.3 Gerência de Configuração

A Gerência de Configuração (GCO) visa manter e disponibilizar produtos de trabalho, além de um controle das alterações feitas no projeto e garantir a integridade do projeto e a execução consistente do que foi planejado. Seu propósito é garantir a integridade dos produtos de trabalho e fornecê-los para todas as pessoas envolvidas com o projeto (GUIA, 2011).

Segundo SOFTEX (2011, p.52) durante o desenvolvimento “o sistema de Gerência de Configuração é fundamental para prover controle sobre os produtos de trabalho produzidos e modificados por diferentes engenheiros de software”. Também disponibiliza um acompanhamento do andamento das tarefas de desenvolvimento.

O processo Gerência de Configuração não orienta como serão feitos os procedimentos de modificações dos produtos, e sim, auxilia nos processos de controle e acompanhamento dessas necessidades de modificações.

Alguns procedimentos são esperado para o processo de Gerência de Configuração do MPS.BR, de forma a garantir os resultados (GCOs) do processo, como é detalhado no Quadro 1.

Resultados esperados	Definição
GCO1 - Um Sistema de Gerência de Configuração é estabelecido e mantido	Esse sistema pode ser decomposto em três subsistemas: sistema de controle de versões, sistema de controle de modificações e sistema de gerenciamento de construção.

GCO2 - Os itens de configuração são identificados com base em critérios estabelecidos	Por meio do plano de Gerência de Configuração a seleção do que será considerado um item de configuração é usualmente baseada em critérios previamente estabelecidos.
GCO3 - Os itens de configuração sujeitos a um controle formal são colocados sob *baseline*	Os itens de configuração identificados no resultado esperado GCO2 serão produzidos de acordo com os momentos previamente estabelecidos. As *baselines* são guias do que foi planejado já com todas ou a maioria dos atributos estabelecidos e aumentam o nível de controle sobre os itens de configuração.
GCO4 - A situação dos itens de configuração e das *baselines* é registrada ao longo do tempo e disponibilizada	A geração e liberação de *baselines* precisam ser registradas e disponibilizadas em um nível de detalhe suficiente para que o conteúdo e a situação de cada item de configuração sejam conhecidos e que versões anteriores possam ser recuperadas.
GCO5 - Modificações em itens de configuração são controladas	A partir do momento que os itens de configuração passam a fazer parte de uma *baseline*, toda e qualquer modificação sobre esses itens de configuração deve passar por um processo formal de controle de modificações.
GCO6 - O armazenamento, o manuseio e a liberação de itens de configuração e *baselines* são controlados	Todos os produtos de trabalho que forem itens de configuração, tanto de projetos quanto de processos, são armazenados no sistema de Gerência de Configuração, seguindo as especificações definidas para cada tipo de item de configuração.
GCO7 - Auditorias de configuração são realizadas objetivamente para assegurar que as *baselines* e os itens de configuração estejam íntegros, completos e consistentes	As auditorias têm o objetivo de verificar se os procedimentos e diretrizes estão sendo seguidos de forma correta e adequada, bem como se os itens de configuração e as *baselines* estão íntegras, corretas e consistentes.

Quadro 1 - Processo de Gerência de Configuração
Fonte: SOFTEX (2011)

2.3 FERRAMENTAS DE GCS

A seguir serão apresentadas algumas ferramentas de GCS existentes e utilizadas por empresas de desenvolvimento de software, de forma a identificar suas características, pontos fortes e fracos.

2.3.1 JIRA + CONFLUENCE

O Jira é uma ferramenta *open source* desenvolvida pela empresa Atlassian, possuindo complementos em versões pagas. Segundo a Atlassian (2014), esta ferramenta é essencialmente para gestão e acompanhamento de projetos, desde as tarefas a serem desenvolvidas, como o controle dos artefatos gerados durante o desenvolvimento. O Jira também implementa a gestão de mudanças sobre os códigos produzidos, com a utilização de *plug-ins* com ferramentas de versionamento.

Esta ferramenta implementa a GCS pelo registro de mudanças exercidas sobre os artefatos controlados, que são análogos aos itens de configuração (IC). Nesta perspectiva os projetos são análogos à *baselines*, que passam pelos mesmos processos definidos pelas GCOs. A ferramenta não implementa um controle de mudanças, qualquer pessoa com acesso ao item pode alterá-lo, sem que haja uma análise de impacto prévia.

É possível integrar ao JIRA outro software de gerenciamento de projetos da empresa Atlassian, que é o Confluence, um ambiente mais abrangente no que diz respeito ao escopo, que permite o gerenciamento de projeto de uma forma mais detalhada. É um ambiente que pode ser usado para o acompanhamento de documentos de um colaborador, cadastro de solicitações de mudanças por parte de clientes, informações institucionais, acompanhamento e criação de indicadores de projeto.

Os ICs que compõe o projeto são armazenados no repositório do Confluence e são utilizados por pendências cadastradas no JIRA, como desenvolvimento, análise ou validação. Desta forma quando uma alteração é feita sobre um IC ou mesmo sobre uma pendência que utiliza um IC, o sistema gera um histórico do mesmo, contendo as alterações efetuadas. Ao gerar um histórico de um IC o sistema notifica os responsáveis pelo mesmo. A rastreabilidade de um IC se dá por meio das referências que o mesmo tem sobre uma pendência e vice-versa, então a cada alteração o sistema verifica o impacto.

2.3.2 SUBVERSION (SVN)

Esta é uma ferramenta *open source* que realiza a gestão de mudanças e versionamento em arquivos por meio do fornecimento de um repositório de armazenamento dos dados comum a todos os clientes que utilizam os documentos. Este repositório é acessado via *Uniform Resource Locator* (*URL*). Nele são armazenados todos os *branchings* – que são vertentes de desenvolvimento a partir das quais serão feitas as modificações – os quais fornecem (*checkout*) e recebem (*checkin*) arquivos. Uma vez que um arquivo esteja sob configuração, o SVN se

encarrega de manter o *branching* com as versões mais atuais dos arquivos que o compõem. Uma vez que uma nova versão de um arquivo é gerada a cada alteração que ele sofre, porém, o SVN armazena um histórico de modificações contendo informações de quem realizou a alteração e da data da alteração, permitindo realizar um comparativo entre as versões e definir uma determinada versão de um arquivo como atual (restaurando-a). Segundo Pereira (2008), o SVN permite dois tipos de abordagem em sua utilização:

- **Cópia-Modificação-Mesclagem:** Nesta abordagem os itens sob controle de versionamento são disponibilizados para alteração a todos os usuários que possuem acesso ao repositório. O usuário deve fazer uma cópia dos dados do repositório, o que é tratado pelo SVN como área de trabalho. A partir do *checkout* dos dados o usuário faz as alterações que deseja e sincroniza o arquivo com o repositório. Nesta sincronização com o repositório o SVN identifica possíveis conflitos – uma vez que nesta abordagem a alteração sobre um arquivo pode ser concorrente, já que ele está disponível para todos os usuários – e permite ao usuário "*comitar"* (fazer *upload* do arquivo para o repositório) e/ou mesclar as alterações com a versão corrente do arquivo. É neste momento que o SVN atualiza o *branching* principal e gera um histórico de alteração do arquivo.
- **Reserva-Modifica-Libera:** Nesta abordagem o SVN permite que apenas um usuário altere um determinado arquivo. Para alterar o arquivo o usuário deve reservá-lo e toda a equipe pode saber para quem está feita a reserva do respectivo arquivo. Esta abordagem evita conflitos de mesclagem, tornando-se mais seguro, porém mais restritivo.

2.3.3 GIT

GIT é uma ferramenta *open source* que permite a gestão de versionamento e mudanças de uma forma semelhante ao SVN, porém, mais descentralizada (GIT,

2014). Foi desenvolvida essencialmente para controlar as versões do *kernel* do Linux, tendo assim, o envolvimento da comunidade de desenvolvimento Linux como projetistas e desenvolvedores. O principal objetivo do desenvolvimento do GIT era sanar os problemas de desempenho que o SVN possuía na época em que o GIT foi desenvolvido (2005), portanto ele possui muitas características do SVN.

O GIT permite gerar repositórios como o SVN, porém, é possível gerar repositórios filhos de um repositório central, ramificando o ambiente de desenvolvimento do projeto, permitindo aos usuários terem áreas de trabalhos ideais para suas necessidades e/ou para cada etapa de desenvolvimento. Para tanto ele fornece formas de propagar mudanças de um repositório para outro, fazendo mesclagens a cada propagação, para eventualmente manter a integridade do ramo principal dos repositórios.

A principal característica dessa ferramenta é a mesclagem inteligente em que a cada atualização feita no servidor o próprio servidor se encarrega de fazer a verificação de conflitos e resolução dos mesmos. Uma grande vantagem em utilizar o GIT ou até mesmo o SVN é a interoperabilidade que eles possuem oferecendo, além da gestão de configuração sobre arquivos em geral, alguns *plug-ins* que podem ser usados durante a codificação de um projeto de software, fazendo a mesma gestão descrita anteriormente, porém, de uma forma integrada ao ambiente de desenvolvimento – *Integrated Development Environment* (IDE).

2.3.4 Conclusões sobre as Ferramentas Estudadas

O GIT e o Subversion são ferramentas com fins de versionamento de código, adaptadas por empresas que desejam implementar a gestão de mudanças de forma manual, e até mesmo parcial, uma vez que não controla o registro de solicitações de mudanças. Já o Jira e o Confluence permitem a implementação da Gestão de Configuração, mas de forma tão minuciosa que sua aplicação em ambientes de empresas de pequeno e médio porte não se torna viável. Quanto à gestão de mudanças, é realizada de forma que é possível identificar o que foi alterado, porém, não é possível recuperar arquivos físicos de versões anteriores.

Portanto, por mais recursos que as ferramentas estudadas disponham, a Gestão de Configuração e Mudanças não é abrangida de forma que os resultados esperados pelas GCOs definidas no nível F do MPS.BR sejam atingidos.

3 MATERIAIS E MÉTODOS

A ênfase deste capítulo está em reportar o que e como será feito para alcançar o objetivo do estágio.

3.1 MATERIAIS

O Quadro 2 apresenta as ferramentas e as tecnologias que foram utilizadas para modelar e implementar o sistema.

Ferramenta / Tecnologia	**Versão**	**Referência**	**Finalidade**
Visual Paradigm Community	11.0	http://www.visual-paradigm.com/solution/freeumltool/	Ferramenta para desenvolvimento de UML, diagramas, etc.
Net Beans	8.0	https://netbeans.org/downloads/	Ambiente de desenvolvimento.
Java EE	7	https://www.java.com/pt_BR	Especificação Java para desenvolvimento de aplicações web.
Prime Faces	4.0	http://primefaces.org	Biblioteca de componentes para desenvolvimento da interface.
JPA/JTA (Hibernate)	4.3	http://hibernate.org	*Framework* para persistência de dados e controle das transações.
JavaServer Faces	2.2	http://www.oracle.com/technetwork/java/javaee/javaserverfaces-139869.html https://javaserverfaces.java.net/	*Framework* para desenvolvimento das interfaces.
MySQL Cluster	5.6	http://www.mysql.com	Sistema gerenciador de banco de dados.
MySQL Workbench	6.0	http://www.mysql.com	Modelagem do diagrama de entidades e relacionamentos no banco de dados.
Glassfish	4.0	https://glassfish.java.net/download.html	Servidor Web de aplicações.
Maven	3.0	http://maven.apache.org/download.cgi	Gerenciador de dependências.

Quadro 2 - Ferramentas e tecnologias utilizadas

3.2 MÉTODO

O processo de desenvolvimento adotado seguiu um modelo iterativo e incremental, uma vez que a necessidade de modificar e/ou adicionar requisitos era encontrada durante o desenvolvimento. Este processo divide o desenvolvimento em ciclos, e em cada ciclo, podem ser identificadas as fases de análise, projeto, implementação e testes (BEZERRA, 2002). Desta maneira o desenvolvimento acontece de uma maneira versionada, ou seja, com incrementos de funcionalidades a cada iteração.

O Quadro 3 apresenta as etapas do processo de desenvolvimento do trabalho em questão e as iterações realizadas.

Iterações / **Etapas do processo**	**1ª iteração**	**2ª iteração**	**3ª iteração**	**4ª iteração**	**5ª iteração**
Levantamento do referencial teórico	Elaboração de referencial teórico envolvendo aspectos de gestão de configuração necessários à elaboração do trabalho				
Levantamento de requisitos	Identificação dos requisitos básicos.	Validação dos requisitos identificados.			
Análise e projeto	Modelagem das classes e tabelas.	Definição e expansão dos casos de uso. Elaboração do modelo conceitual e ajustes na modelagem.	Ajustes na modelagem	Ajustes na modelagem	
Implementação		Implementação de padrões de telas	Validação e implementação de padrões de telas e primeiras funcionalidades	Implementação de todas as funcionalidades do sistema e realização de testes unitários	Implementação das correções de defeitos nas funcionalidades e também ajuste do *layout* final. funcionalidades
Testes			Testes das primeiras funcionalidades	Realização dos testes de integração.	Validação das correções.

Quadro 3 - Iterações definidas

A seguir estão descritas as etapas apresentadas no Quadro 3 definidas para o desenvolvimento desse trabalho e as principais atividades realizadas em cada etapa:

a) **Levantamento do referencial teórico:** O referencial teórico abrangeu o conceito geral de gestão de configuração, tendo ênfase no modelo definido pelo MPS.BR. A intenção no levantamento do referencial teórico foi compreender o processo de GCS e buscar possíveis soluções para os problemas encontrados nas ferramentas que aplicam a GCS disponíveis no mercado, identificando processos comuns nos diversos modelos de GCS, bem como suas deficiências. A partir da definição das necessidades de desenvolvimento foram identificadas as tecnologias a serem utilizadas, como foco tanto em desempenho quanto em interface, por isso foi escolhido o JSF implementado com o *PrimeFaces*.

b) **Levantamento de requisitos:** Primeiramente foi feita a definição do escopo do projeto. A partir da definição do escopo do projeto, foram levantadas as necessidades da ferramenta desenvolvida sob a perspectiva do ambiente de aplicação, ou seja, considerando sua utilização em uma fábrica de software para ser utilizada na disciplina de Oficina de Projeto e Desenvolvimento de Software. O contexto dessa fábrica foi proposto pelas professoras Beatriz Terezinha Borsoi, Eliane De Bortoli Fávero e Lucilia Yoshi Araki com o objetivo de representar seus requisitos essenciais. Destaca-se que o projeto aqui proposto será integrado a um módulo de Gestão de Projetos já existente, o qual também foi desenvolvido considerando o contexto de uma fábrica de software, conforme citado anteriormente. Um diagrama de atividades foi desenvolvido a fim de representar o processo de gerência de configuração a ser adotado para esse projeto.

c) **Análise e projeto do sistema:** A partir da definição dos requisitos foram definidos e expandidos os casos de uso que seriam desenvolvidos, elaborado o modelo conceitual e definidas e mapeadas as entidades envolvidas, bem como as tabelas necessárias.

d) Implementação: Foi criado um novo repositório utilizando a ferramenta SVN, para armazenamento e controle de versões do desenvolvimento. A partir da definição das relações do sistema, foram construídas e mapeadas as entidades utilizando *JPA Hibernate*. A partir do projeto existente para gestão da fábrica de software, conforme já citado, foi feita uma separação do espaço de trabalho no ambiente do sistema, separando o módulo de gestão de projetos (já desenvolvido) do módulo de gestão de configuração abrangido pelo sistema em questão, além disso, foram adicionadas as funcionalidades apresentadas nos casos de uso definidos neste documento. A IDE utilizada para o desenvolvimento foi o *NetBeans*. Os *layouts* das telas ficaram com o padrão de interfaces provido pelos componentes da biblioteca *PrimeFaces*, salvos ajustes de alinhamento e diagramação feitos por *Cascading Style Sheets* (CSS), ou Folhas de Estilo em Cascata.

e) Testes: Os testes foram segmentados entre: testes por caso de uso (unitários) e testes de integração. Os testes por caso de uso foram efetuados à medida em que os mesmos ficavam prontos, já o teste de integração foi realizado após a conclusão de todos os casos de uso. Defeitos encontrados em ambos segmentos de testes foram corrigidos no momento em que foram detectados, porém, não foram registrados.

4 RESULTADOS E DISCUSSÕES

Este capítulo apresenta os resultados do desenvolvimento deste trabalho que consistiu na modelagem e implementação de um sistema para gestão de configuração em projetos de software, o qual foi desenvolvido considerando o contexto de uma fábrica de software a ser simulada em uma disciplina de Oficina de Desenvolvimento de Software, destacando-se a gestão de IC e *baselines,* gestão de mudanças e versionamento dos ICs que as compõe.

4.1 ESCOPO DO SISTEMA

O sistema proposto visa prover a gerência da configuração de projetos de software, implementando os procedimentos definidos no nível F do MPS.BR, nas respectivas GCOs (SOFTEX, 2011). O sistema desenvolvido permite o controle de *baselines* de um projeto, desde sua concepção à sua liberação ao cliente.

Os artefatos que fizerem parte da composição de uma *baseline* passarão a ser considerados itens de configuração e por consequência a estarem sobre controle de versão. Este controle de versão deve ocorrer por meio do registro de modificações necessárias aos itens de configuração. Para cada modificação será realizada uma análise de impacto, identificando a necessidade de alteração da versão do item de configuração. A mudança de versão deve ocorrer de forma manual, ou seja, o usuário deve decidir sobre sua alteração ou não, a partir da análise de impacto. O sistema deve manter um histórico de versões para cada item de configuração, permitindo a recuperação de versões anteriores, caso necessário.

O aplicativo desenvolvido a partir desse trabalho visa ser utilizado como complemento ao sistema de gestão de projetos da disciplina de Oficina de Projeto e Desenvolvimento de Software, em ambientes reais se destina ao gerenciamento de fábricas de software definidas como um conjunto de processos integrados e elaborados de acordo com modelos de qualidade. A sua utilização em disciplinas acadêmicas visa facilitar o trabalho do professor no gerenciamento das atividades e

das equipes de projeto (os alunos da disciplina). Destaca-se que o sistema em questão não fará o controle de acesso aos itens de configuração e nem mesmo de artefatos de código.

4.2 MODELAGEM DO SISTEMA

Esta Seção apresenta os resultados previstos para esse trabalho, consistindo da modelagem de um sistema para gestão de configuração em projetos de desenvolvimento de software, mais especificamente da disciplina de Oficina Projeto e Desenvolvimento de Software. Na próxima Seção o sistema implementado, consistindo dos códigos e telas gerados.

4.2.1 Análise e Projeto do Sistema

A Figura 3 apresenta um diagrama de atividades contendo o processo de gestão de configuração no qual se baseia o sistema desenvolvido nesse trabalho.

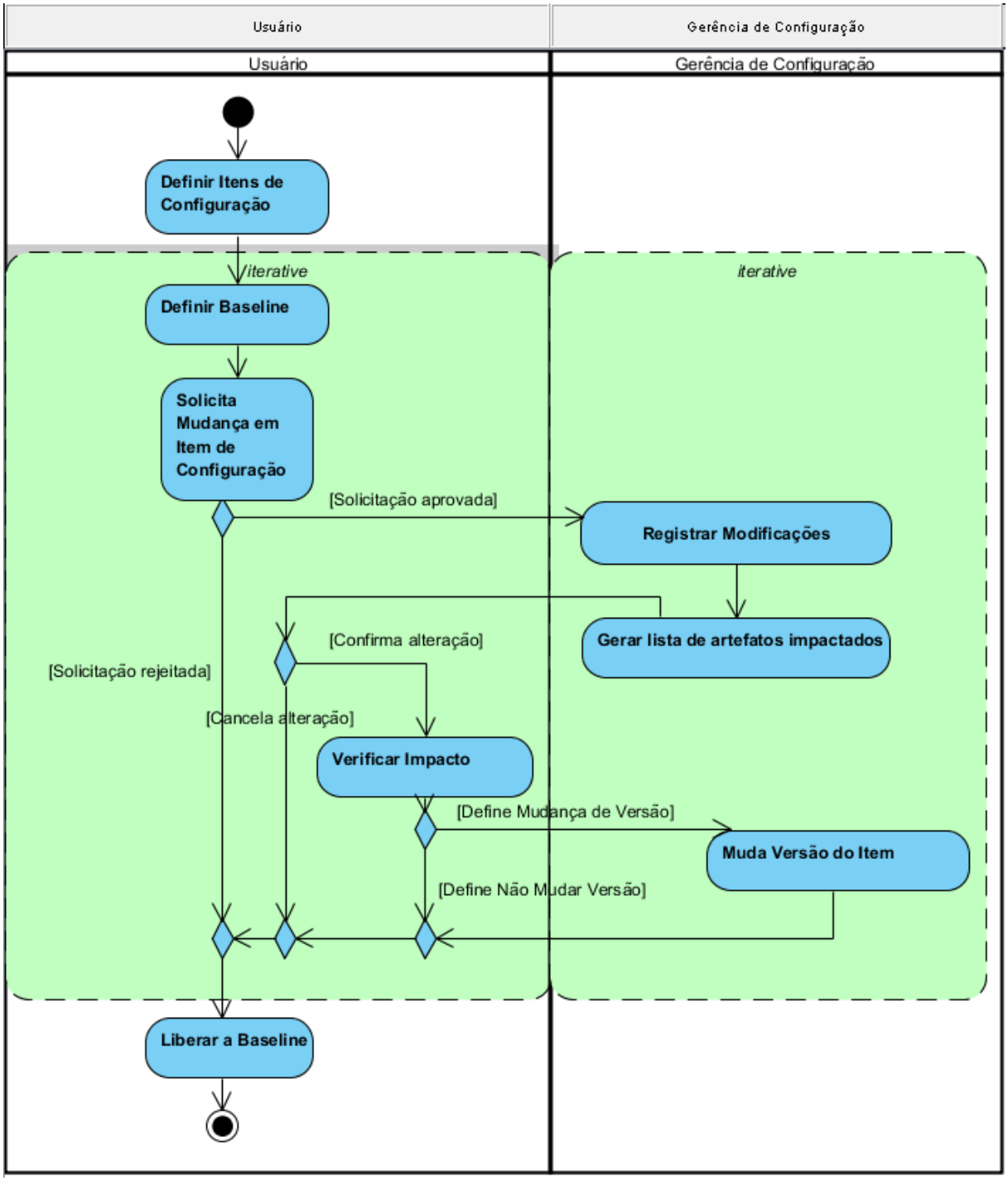

Figura 2 - Modelo do processo de gestão de configuração

Os Quadros 4 até 10 apresentam os requisitos funcionais e não funcionais para o sistema.

F1 Definir *baseline*	**Oculto ( )**
Descrição: O sistema deve registrar uma *baseline*, permitindo a inclusão de múltiplos itens de configuração que farão parte de sua composição.	

Quadro 4 - Defiinir *baseline*

F2 Registrar solicitação de modificação	**Oculto ( )**
Descrição: O sistema deve permitir registrar as solicitações e modificações feitas em itens de configuração que componham uma determinada *baseline.* Esta modificação será efetivada a partir da confirmação de uma solicitação de modificação pelo usuário responsável.	

Quadro 5 - Registrar solicitação de modificação

F3 Registrar modificação em IC	**Oculto ( )**
Descrição: O sistema deve permitir registrar as solicitações e modificações feitas em itens de configuração que componham uma determinada *baseline.* Esta modificação será efetivada a partir da confirmação de uma solicitação de modificação pelo usuário responsável.	

Quadro 6 – Registrar modificação em IC

F4 Analisar o impacto das modificações			**Oculto ( )**	
Descrição: O sistema deve permitir fazer a análise de impacto das modificações realizadas a partir da visualização dos artefatos relacionados, permitindo a mudança ou não de versão do artefato modificado de forma manual.				
Requisitos Não-Funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
NF5.1 Visualização dos artefatos impactados	A visualização dos artefatos impactados deve ser feita com base na matriz de rastreabilidade de artefatos já existente.	Regra de negócio	( )	(X)

Quadro 7 - Analisar o impacto das modificações

F5 Armazenar histórico de versões de itens de configuração	**Oculto (x)**
Descrição: O sistema deve permitir o armazenamento das diversas versões de itens de configuração, sempre deixando disponível a última versão.	

Quadro 8 - Armazenar histórico de versões de itens de configuração

F6 Liberar *baseline*	**Oculto ( )**
Descrição: O sistema deve permitir a realização de entregas de *baseline*s. O usuário selecionará uma determinada *baseline* e a enviará para o receptor desejado.	

Quadro 9 - Liberar *baseline*

F7 Gerar relatórios de modificações	**Oculto ( )**
Descrição: O sistema deve permitir a geração de relatórios das modificações realizadas em itens de configuração considerando aspectos como a fase em que ocorreram as mudanças, esforço para a realização da mudança, mudanças ocorridas em cada versão do arterfato, entre outros.	

Quadro 10 - Gerar relatório de modificações

A Figura 4 apresenta o modelo de casos de uso para o sistema proposto.

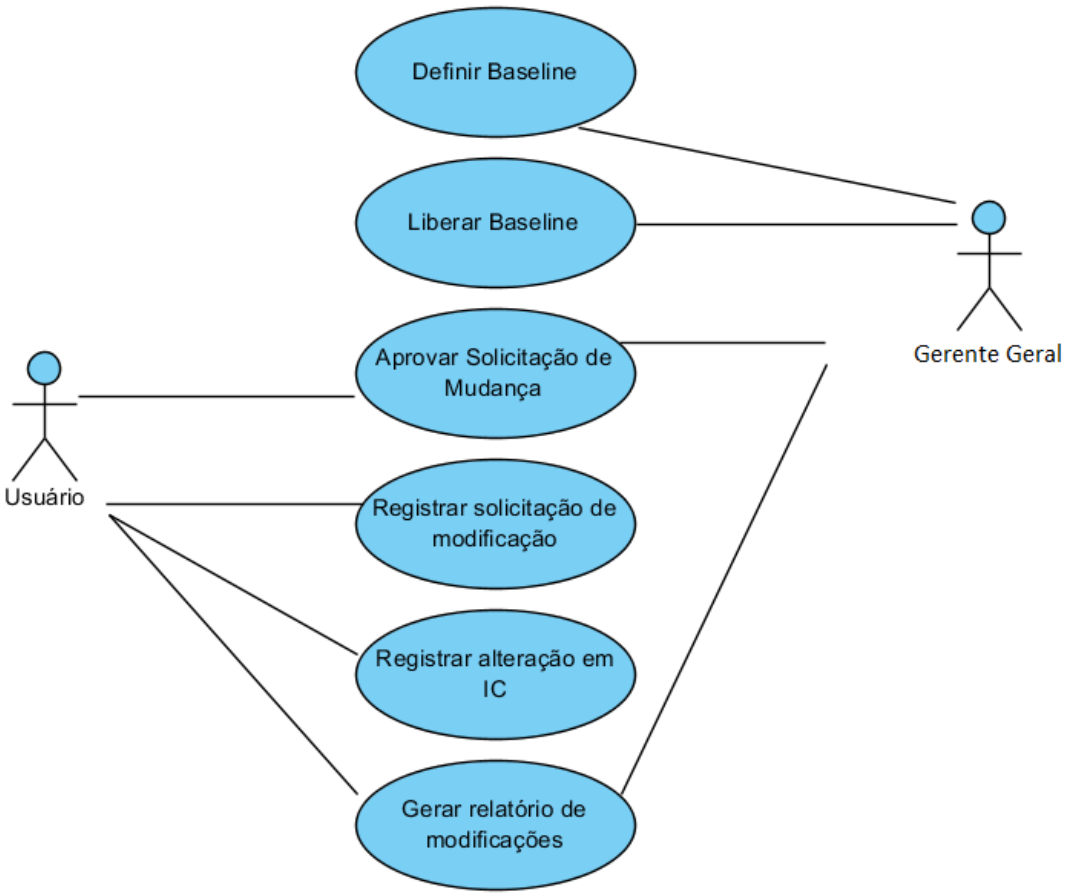

Figura 3 - Modelo de casos de uso para Gestão de Configuração

O Quadro 11 apresenta a expansão do caso de uso Definir *baseline*, a qual a apresenta a especificação detalhada da realização do processo.

Caso de Uso: UC1 - Definir *baseline*
Atores: Gerente de projeto
Precondições: O usuário deve possuir uma conta no sistema. Login realizado com sucesso. Ao menos um Item de Configuração cadastrado.
Pós-condições: *Baseline* cadastrada com sucesso. Inclusão/Alteração cancelada.
Sequência típica de eventos (Fluxo Principal): Esse caso de uso inicia quando o usuário solicita a inclusão de uma *baseline*: 1. [IN] O usuário seleciona a opção “Definir *Baseline*”. 2. [IN] O usuário informa os atributos do cadastro da *Baseline* e confirma o cadastro. (1 exceção) 3. [OUT] O sistema emite uma mensagem de aprovação da ação. 4. [OUT] O sistema inclu o registro da *baseline* e finaliza o caso de uso.
Tratamento de Exceções e Variantes: **Exceção 2a: O usuário não adiciona nenhum Item de Configuração à *baseline*.**

Variante 2a.1: O usuário adiciona um ou mais Itens de Configuração. 2a.1: [IN] O usuário adiciona um ou mais Itens de Configuração. 2a.1.1 [OUT] Retorna ao passo 3. **Variante 2a.2: O usuário desiste da ação.** 2a.2: [IN] O usuário cancela a ação. 2a.2.1 [OUT] O sistema retorna uma mensagem de aviso ao usuário. 2a.2.2 [OUT] O caso de uso é finalizado.

Quadro 11 - UC1 – Definir *baseline*

O Quadro 12 apresenta o caso de uso de registro de solicitação de mudança em um item de configuração.

Caso de Uso: UC2 – Registrar solicitação de modificações
Atores: Usuário
Precondições: O usuário deve possuir uma conta no sistema. Login realizado com sucesso.
Pós-condições: Registro de solicitação de mudança realizado com sucesso.
Sequência típica de eventos (Fluxo Principal): Esse caso de uso inicia quando o usuário deseja solicitar a mudança de um Item de Configuração: 1. [IN] O usuário seleciona a opção “Incluir Solicitação de Modificação”. 2. [IN] O usuário informa os campos do cadastro da solicitação de mudança do IC a ser modificado. 3. [OUT] O sistema emite uma mensagem de aprovação da ação. 4. [OUT] O sistema inclui o registro da solicitação de mudança e finaliza o caso de uso.
Tratamento de Exceções e Variantes: **Exceção 2a: O usuário não preenche todos os campos obrigatórios.** **Variante 2a.1: O usuário preenche todos os campos obrigatórios.** 2a.1.1 Retorna ao passo 3. **Variante 2a.2: O usuário desiste da ação.** 2a.2.1 O sistema retorna uma mensagem de aviso ao usuário. 2a.2.2 O caso de uso é finalizado. **Exceção 2b: Já existe uma solicitação de mudança não aprovada para o mesmo IC.** 2b.1 [[OUT] O sistema retorna uma mensagem de aviso ao usuário. 2b.2 [IN] O caso de uso é finalizado.

Quadro 12 – UC2 - Controlar modificações

O Quadro 13 apresenta o caso de uso de registro de alteração em um IC cuja solicitação de mudança foi aceita.

Caso de Uso: UC3 – Registrar alteração em IC
Atores: Membro da equipe (responsável pela alteração do IC) e o usuário no caso de uso.
Precondições: O usuário deve possuir uma conta no sistema. Login realizado com sucesso.
Pós-condições: Registro de alteração realizado com sucesso.
Sequência típica de eventos (Fluxo Principal):

Esse caso de uso inicia quando um membro da equipe deseja registrar a mudança realizada em um IC: 1. [IN] O membro da equipe registra os dados da alteração. 2. [OUT] O sistema verifica o impacto da alteração. 3. [OUT] O sistema gera uma lista de IC impactados pela alteração e finaliza o caso de uso.
Tratamento de Exceções e Variantes:

Quadro 13 – UC3 – Registrar alteração em IC

O Quadro 14 apresenta o caso de uso de registro de alteração de versão em um IC.

Caso de Uso: UC4 – Aprovar Solicitação de Mudança
Atores: Gerente de projeto ou Usuário
Precondições: Item de Configuração alterado.
Pós-condições: Item de Configuração têm versão alterada. Item de configuração permanece com a mesma versão.
Sequência típica de eventos (Fluxo Principal): Esse caso de uso inicia quando o usuário altera um Item de Configuração e o sistema gera uma lista de IC impactados pela alteração. 1. [IN] O gerente de projeto analisa o impacto da alteração do IC, a partir da lista de itens impactados gerada pelo sistema e confirma a mudança de versão. 2. [OUT] Se a versão do IC mudar o sistema irá armazenar o estado anterior à alteração, como um histórico. 3. [OUT] O sistema emite uma mensagem de aprovação da ação. 4. [OUT] O sistema verifica o impacto da alteração do item e altera a versão do mesmo.

Quadro 14 – UC4– Verificar mudança de versão

O Quadro 15 apresenta o caso de uso de liberação de *baseline*.

Caso de Uso: UC5 - Liberar *baseline*
Atores: Gerente de projeto
Precondições: O usuário deve possuir uma conta no sistema. Login realizado com sucesso. Ao menos uma *baseline* cadastrada.
Pós-condições: Liberação efetuada com sucesso. Liberação cancelada.
Sequência típica de eventos (Fluxo Principal): Esse caso de uso inicia quando o gerente de projeto solicita a liberação de uma *baseline*: 1. [IN] O usuário seleciona uma *baseline* e aciona a opção “Liberar *Baseline”*. 2. [IN] O usuário informa os campos da liberação da *baseline*. 3. [OUT] O sistema encaminha a *baseline* para os destinatários informados.

4. [OUT] O sistema altera o *status* da *baseline* para “Liberada” e finaliza o caso de uso.
Tratamento de Exceções e Variantes: **Exceção 2a: O usuário não preenche todos os campos obrigatórios.** **Variante 2a.1: Os campos faltantes são preenchidos** 2a.1.1 [IN] O usuário preenche os campos obrigatórios. 2a.1.2 [OUT] Retorna ao passo 3. **Variante 2a.2: O usuário desiste da ação.** 2a.2.1 [OUT] O sistema retorna uma mensagem de aviso ao usuário. 2a.2.2 [OUT] O caso de uso é finalizado.

Quadro 15 - UC5 – Liberar *baseline*

A Figura 4 apresenta o diagrama de entidade e relacionamento para o sistema proposto. As cores rosa, laranja e verde representam respectivamente atributos e tabelas mantidos, alterados e adicionados ao sistema de gestão de projetos existente, a fim de possibilitar a implementação da gestão de configuração.

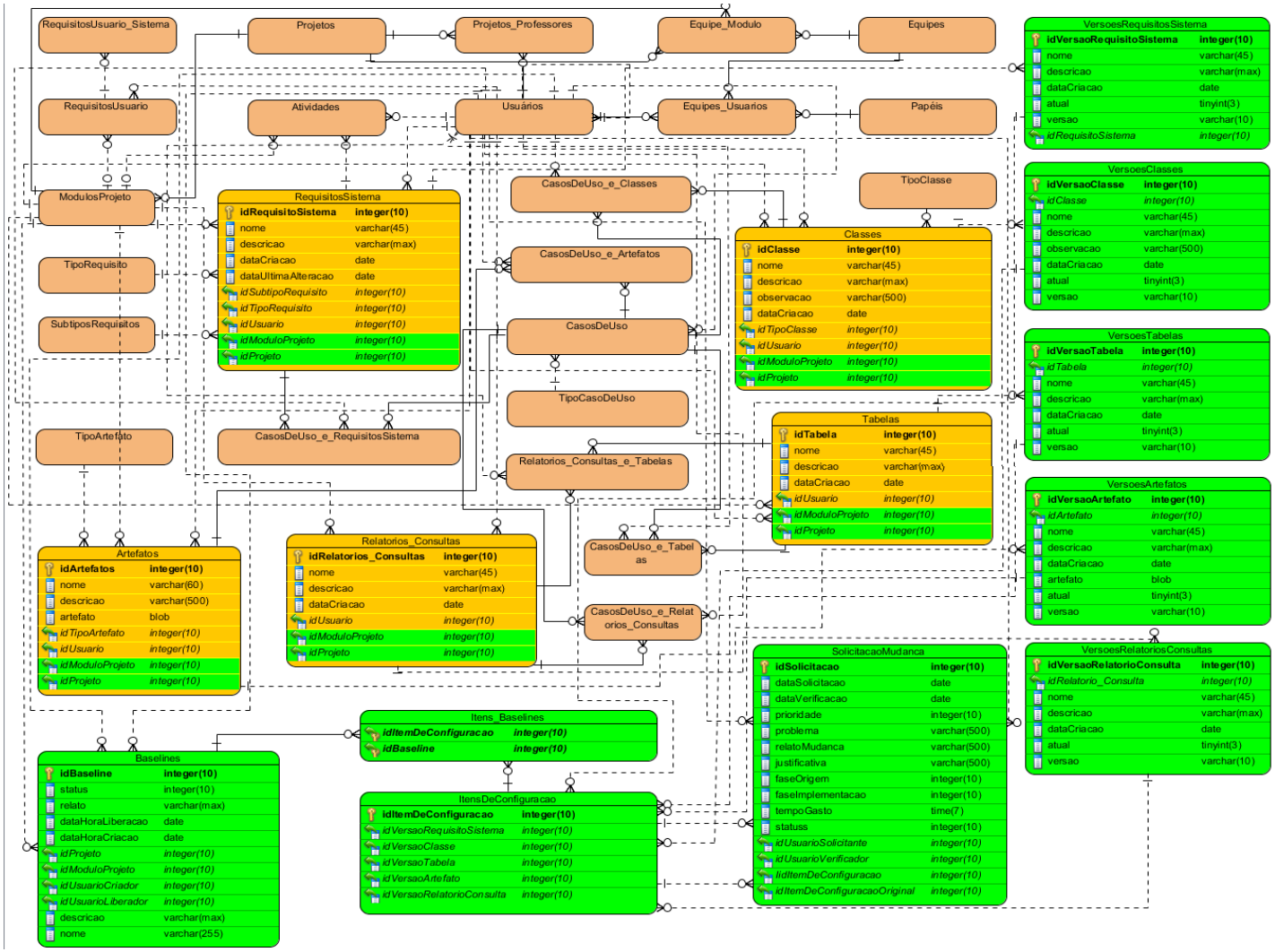

Figura 4 - Diagrama de Entidade e Relacionamento para Gestão de Configuração

A seguir são expandidas as novas tabelas, inerentes à Gestão de Configuração, como apresentadas na Figura 4.

A Tabela 1 apresenta a relação *Baseline*, a qual é composta por IC's. A tabela armazena inicialmente o usuário criador, data da criação e o módulo ao qual a *baseline* pertence. O usuário liberador e o relato de liberação são preenchidos quando o usuário liberar a *baseline*, que consequentemente alterará o *status* da mesma.

Baselines					
Campo	**Tipo**	**Nulo**	**Chave primária**	**Chave estrangeira**	**Observações**
id*Baseline*	Numérico	Não	Sim	Não	
nome	Texto	Não	Não	Não	Identifica uma *baseline* de forma textual.
descricao	Texto	Não	Não	Não	Texto utilizado para descrever as características de uma *baseline*, deixando mais claras as várias instâncias que um projeto ou módulo podem ter.
status	Numérico	Não	Não	Não	Identifica se a *baseline* foi liberada, está em análise ou em edição.
relato	Texto	Sim	Não	Não	Texto utilizado para informar dados pertinentes à liberação de uma *baseline*.
dataHoraLiberacao	Data	Sim	Não	Não	
dataHoraCriacao	Data	Não	Não	Não	
idModuloProjeto	Numérico	Não	Não	Sim	Identifica o módulo e projeto aos quais a *baseline* pertence.
idUsuarioCriador	Numérico	Não	Não	Sim	
idUsuarioLiberador	Numérico	Sim	Não	Sim	

Tabela 1 – *Baselines*

A Tabela 2 apresenta a relação Item de Configuração. Esta tabela representa todos os tipos de artefatos utilizados e/ou gerados durante o processo de desenvolvimento de um projeto de software que compõe uma determinada *baseline*. Um IC equivale a um único artefato, o qual passará a estar sob gestão de mudanças. Os ICs representam artefatos cadastrados nas tabelas RequisitosSistema, Relatorios_Consultas, Classes, Tabelas e Artefatos.

ItensdeConfiguracao					
Campo	**Tipo**	**Nulo**	**Chave primária**	**Chave estrangeira**	**Observações**
idItemDeConfiguracao	Numérico	Não	Sim	Não	
idArtefatos	Numérico	Sim	Não	Sim	
idRequisitoSistema	Numérico	Sim	Não	Sim	
idRelatorios_Consultas	Numérico	Sim	Não	Sim	
idTabela	Numérico	Sim	Não	Sim	
idClasse	Numérico	Sim	Não	Sim	

Tabela 2 - Itens de Configuração

A Tabela 3 representa a efetiva composição da *baseline*, por IC's. Sendo possível que um IC esteja em mais de uma *baseline*.

Itens_Baselines					
Campo	**Tipo**	**Nulo**	**Chave primária**	**Chave estrangeira**	**Observações**
id*Baseline*	Numérico	Não	Sim	Sim	
idItemDeConfiguracao	Numérico	Não	Sim	Sim	

Tabela 3 - Itens da *Baseline*

A Tabela 4 apresenta a relação de Solicitação de Modificação. Esta relação armazena o IC cujo artefato será alterado e ainda armazena as alterações feitas. Caso a solicitação seja aprovada, o sistema permitirá registrar as alterações realizadas e a versão do artefato (conforme definição do usuário após a análise de impacto), alterando também à referência do IC na *baseline*.

SolicitacaoMudanca					
Campo	**Tipo**	**Nulo**	**Chave primária**	**Chave estrangeira**	**Observações**
idSolicitacao	Numérico	Não	Sim	Não	
dataSolicitacao	Data	Não	Não	Não	
dataVerificacao	Data	Sim	Não	Não	
prioridade	Numérico	Não	Não	Não	Indica qual a prioridade de verificação da solicitação.
problema	Texto	Não	Não	Não	Problema que gerou a necessidade de alteração.
justificativa	Texto	Sim	Não	Não	Justificava pela aprovação ou reprovação da solicitação.

faseOrigem	Numérico	Não	Não	Não	Representa a fase em que a solicitação foi criada.
faseImplementacao	Numérico	Não	Não	Não	Representa a fase em que a solicitação foi implementada.
tempoGasto	Time	Sim	Não	Não	Representa o tempo gasto do levantamento à resolução.
status	Numérico	Não	Não	Não	
idUsuarioSolicitante	Numérico	Não	Não	Sim	
idUsuarioVerificador	Numérico	Sim	Não	Sim	
idItemDeConfiguracaoAtual	Numérico	Não	Não	Sim	
idItemDeConfiguracao	Numérico	Não	Não	Sim	

Tabela 4 - Solicitação de Mudança

4.3 APRESENTAÇÃO DO SISTEMA

O sistema proposto foi desenvolvido a partir dos resultados da análise e projeto deste trabalho, sendo que foram implementados padrões de telas para análise e conhecimento da arquitetura existente e das tecnologias a serem utilizadas. O projeto foi organizado utilizando a arquitetura *Model View Control* (MVC), separando a camada de apresentação (*view)* da camada de negócio (*control*) e manipulação de dados (*model*), como mostra a Figura 5.

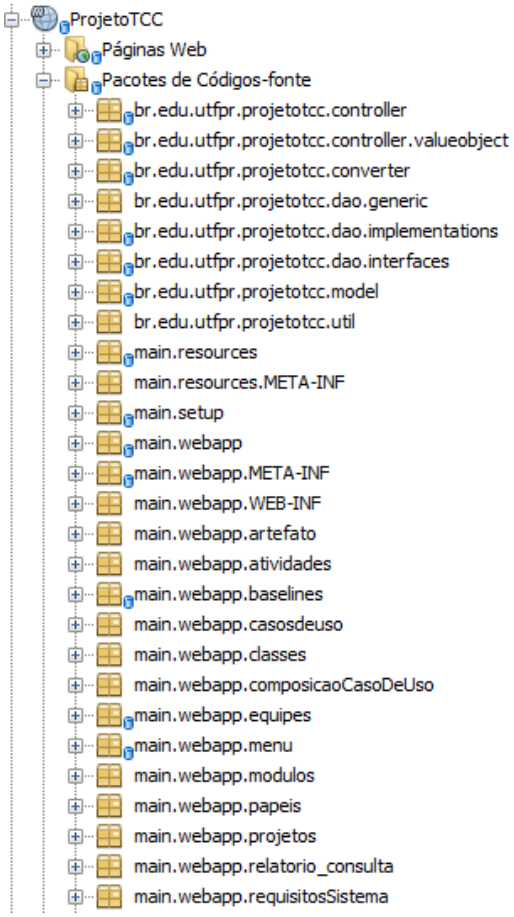

Figura 5 - Aplicação do MVC

A Figura 6 apresenta o menu lateral após a segmentação do sistema da Oficina de Projeto e Desenvolvimento de Software. O agrupamento Gestão de Configuração contém as funcionalidades desenvolvidas neste trabalho.

Figura 6 - Menu Lateral

A Figura 7 apresenta a tela inicial do cadastro de *baselines*, a qual, por padrão, é apresentada como uma listagem. Essa tela é apresentada no formato de uma tabela (*grid*) com linhas selecionáveis, que ao receberem foco permitem que o usuário as libere, exclua ou visualize. Apenas *baselines* que estão aguardando liberação podem ser liberadas ou excluídas. O *link* Atualizar, localizado no canto superior direito, ao ser clicado, atualiza a listagem de *baselines*.

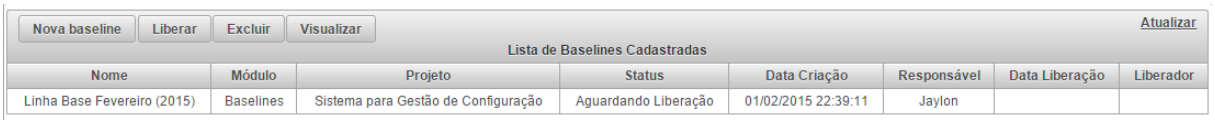

Figura 7 - Tela de listagem de *baselines*

A Figura 8 apresenta a tela de visualização de uma *baseline* que representa a instância do mês de fevereiro do módulo de Baselines do projeto Sistema para

Gestão de Configuração, na qual são apresentados todos os dados relacionados, permitindo ao usuário saber quais itens compõem esta instância.

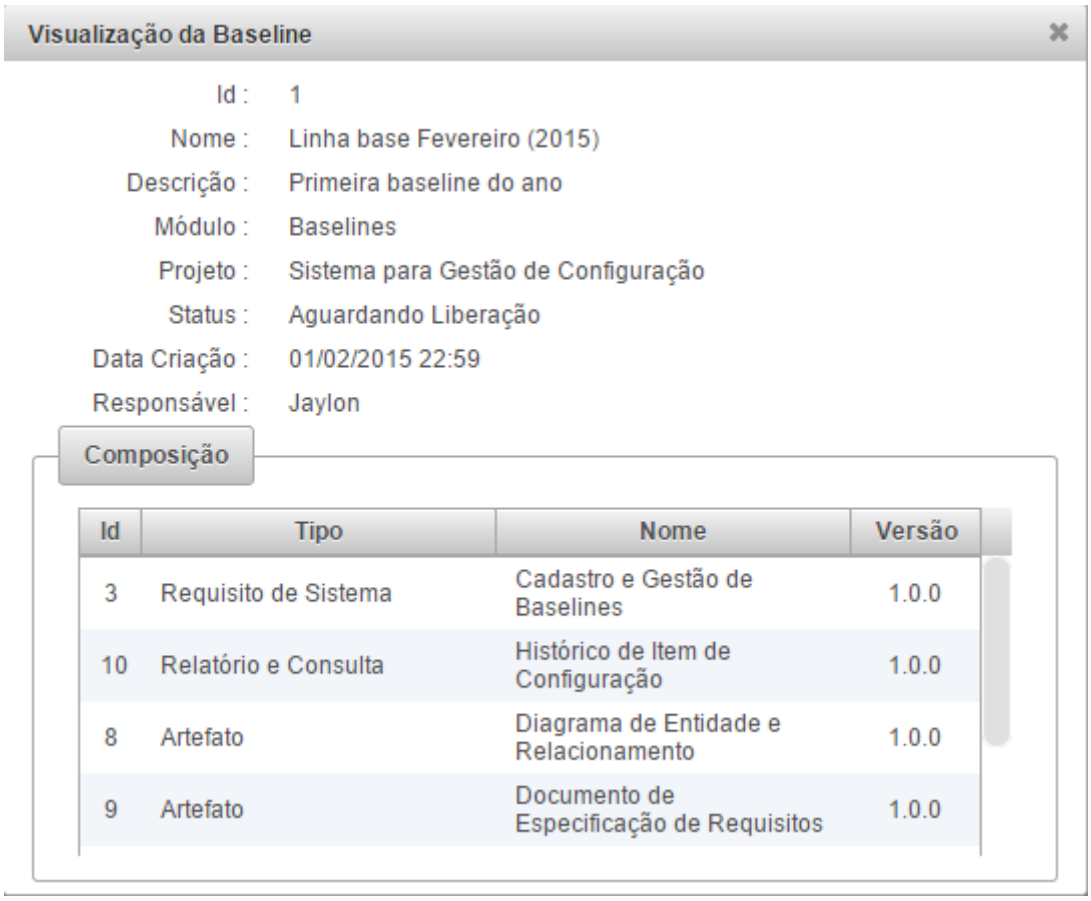

Figura 8 - Tela de visualização da *baseline*

A Figura 9 apresenta a tela de cadastro de *baselines*. O campo de seleção de módulo é preenchido conforme o projeto selecionado. A partir da seleção de um módulo de um projeto, os artefatos disponíveis para a composição da *baseline* serão apenas aqueles pertencentes ao módulo de projeto selecionado. A partir do momento em que um artefato é incluído na composição de uma *baseline,* o sistema o tornará automaticamente um IC, passando a estar sob controle de mudanças. Também é possível adicionar a uma *baseline* um artefato que já tenha sido adicionado em outra *baseline*. As *baselines* de um projeto estarão sempre acessíveis aos gerentes de seus respectivos projetos.

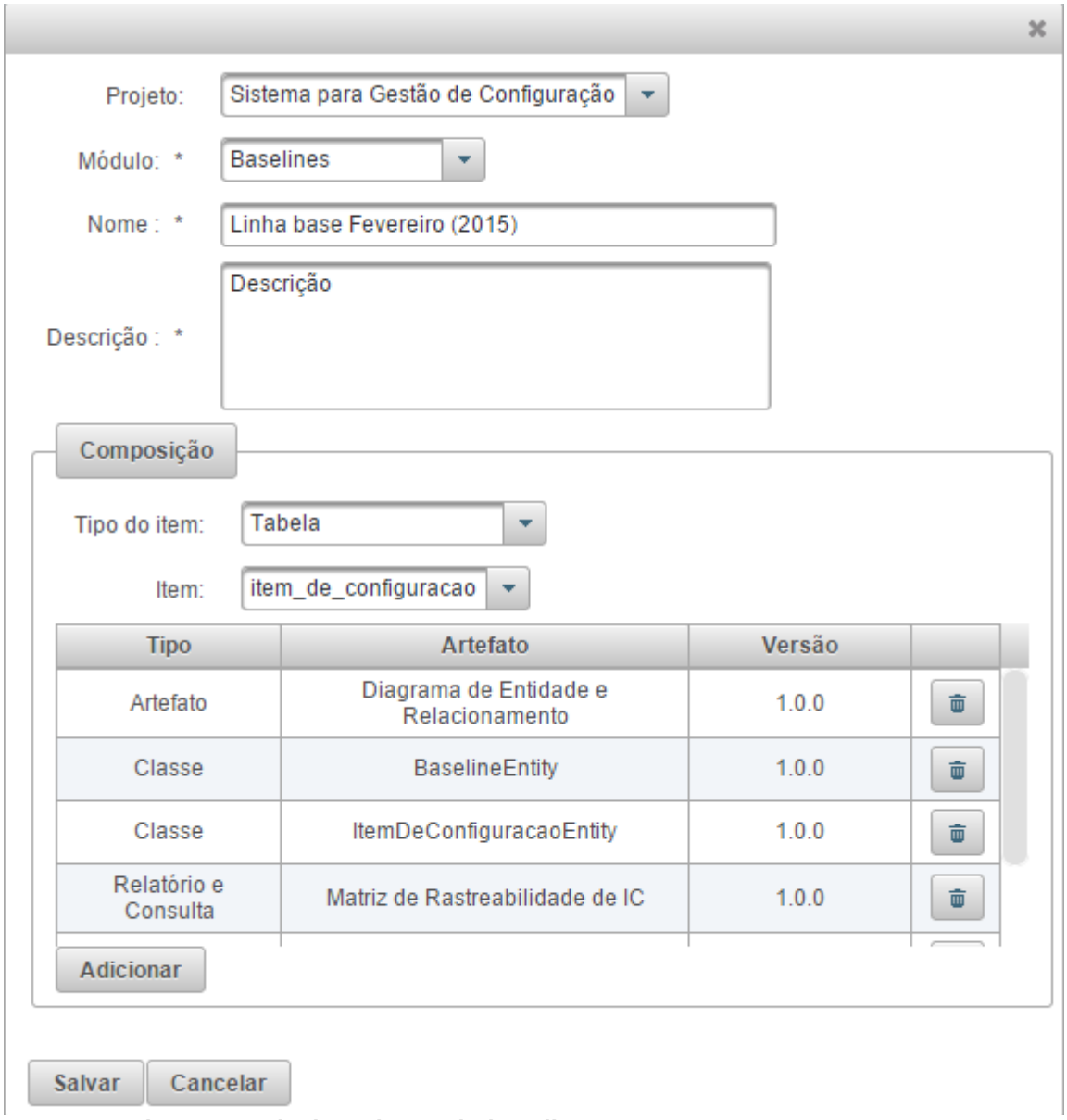

Figura 9 - Tela de cadastro de *baselines*

A Figura 10 apresenta a tela de histórico de artefatos, onde o usuário pode acessar a lista de versões de um determinado item de configuração ou mesmo de um grupo de itens de configuração para um módulo e/ou projeto específico. A partir desta tela o usuário também pode alterar o número da versão de um artefato, bastando para isto selecionar a versão atual do artefato e clicar no botão “Alterar Versão”. Nesta tela o usuário pode visualizar e/ou recuperar os dados/arquivos de versões específicas de um artefato, clicando no botão exibido na última coluna da tabela de listagem.

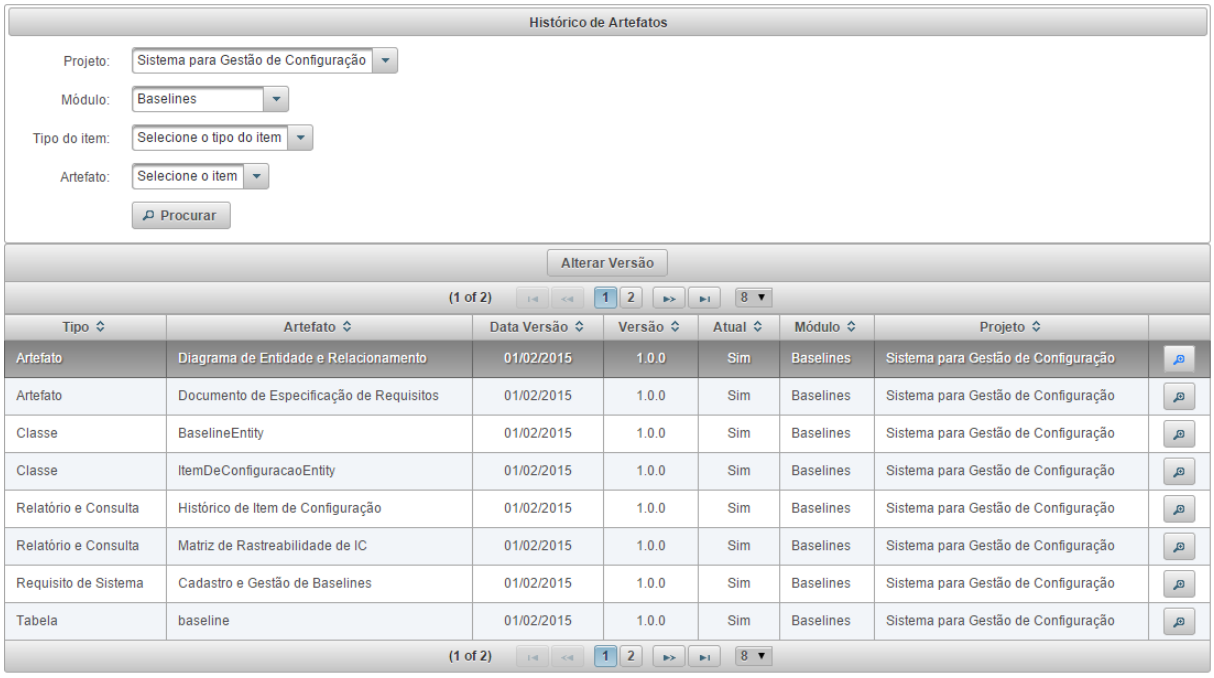

Figura 10 - Histórico de artefatos

A Figura 11 apresenta o mapa de impacto de um artefato, para que a partir deste o usuário possa identificar e/ou justificar a necessidade de alterar a versão de um artefato. O primeiro nível deste mapa de impacto indica o item de configuração que sofreu uma alteração, o segundo nível indica os casos de uso que foram impactados diretamente, o terceiro nível indica os demais itens de configuração impactados indiretamente, e assim sucessivamente.

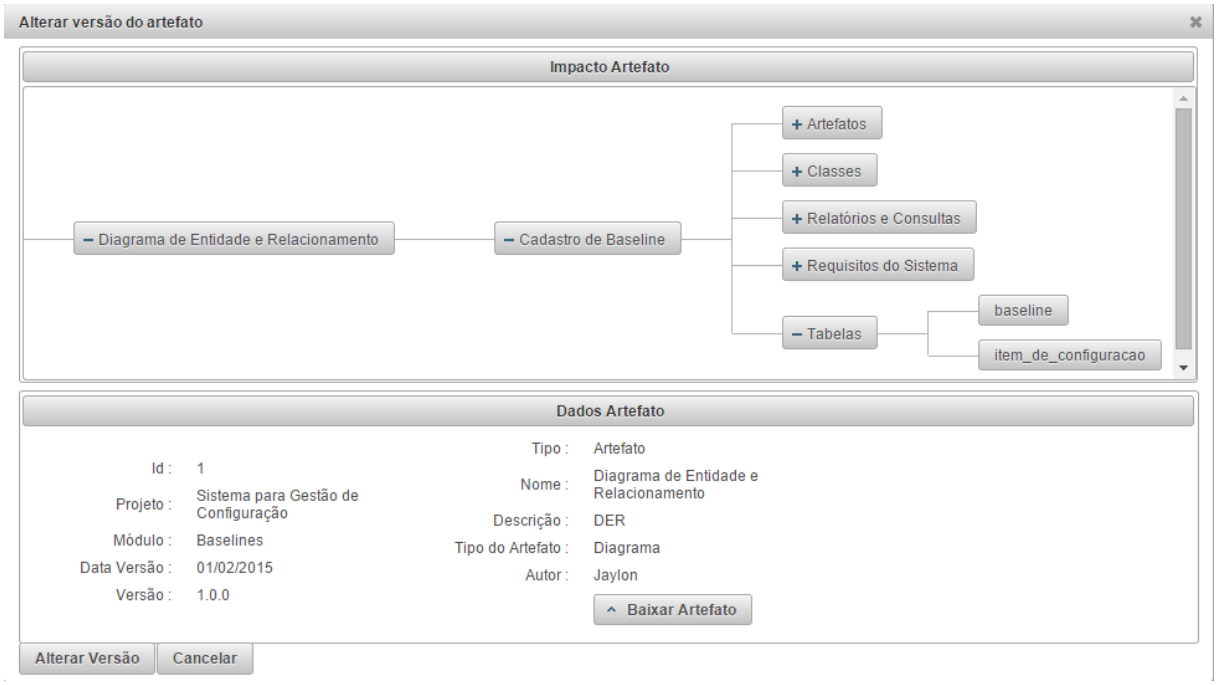

Figura 11 - Mapa de Impacto

A Figura 12 apresenta a tela de visualização de solicitações de mudança, a qual é acessada pela tela de listagem de solicitações de mudança. Nesta tela o usuário responsável por responder à solicitação encontra um comparativo entre a versão original de um artefato e as mudanças requisitadas. A partir deste comparativo o usuário decide se quer ou não aprovar a solicitação, se aprovada o usuário tem a possibilidade de verificar o impacto da mudança e alterar a versão do artefato, como mostrado na Figura 11.

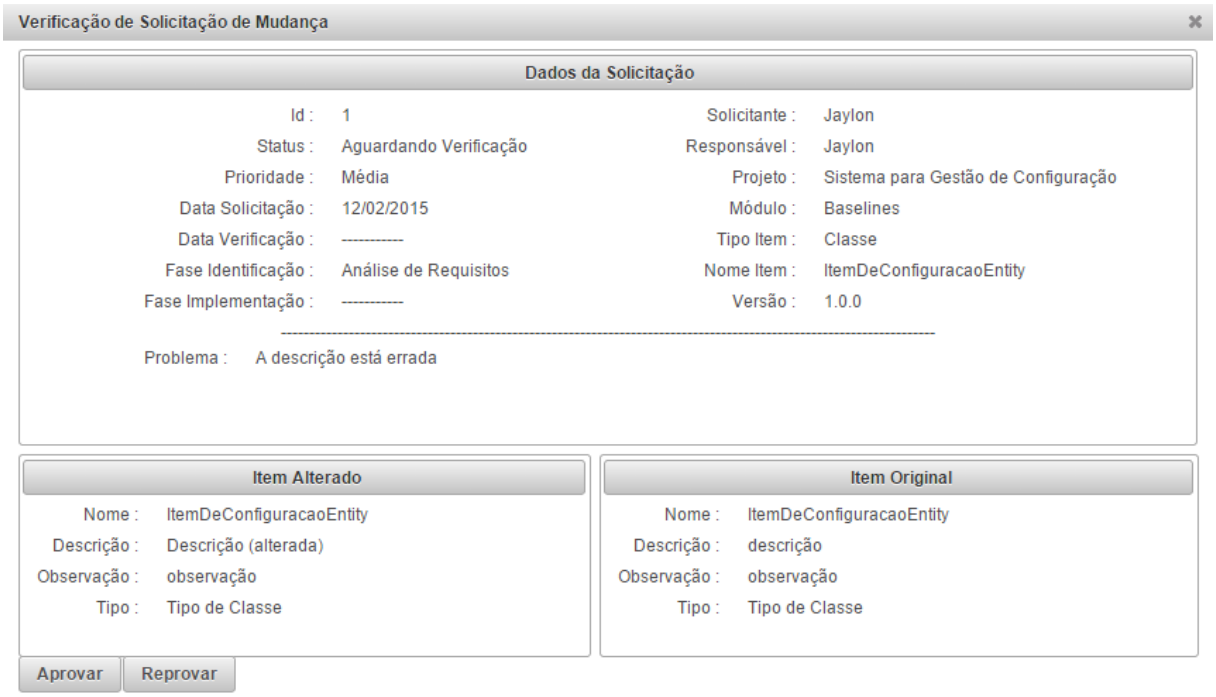

Figura 12 - Verificar Solicitação de Mudança

4.4 IMPLEMENTAÇÃO DO SISTEMA

Seguindo a estrutura existente para construção das interfaces (*views)*, a estrutura das interfaces se dá por uma página agrupadora do conteúdo que envolve as operações *Create*, *Read*, *Update* e *Delete* (CRUD), respectivamente: criação, leitura, atualização e deleção) do cadastro, como mostra a Listagem 1.

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://xmlns.jcp.org/jsf/html"
    xmlns:ui="http://java.sun.com/jsf/facelets">

  <ui:include src="/solicitacoesDeMudanca/list.xhtml"/>
  <ui:include src="/solicitacoesDeMudanca/view.xhtml"/>
  <ui:include src="/solicitacoesDeMudanca/edit.xhtml"/>
</ui:composition>
```

Listagem 1 - Página agrupadora do CRUD de *solicitações de mudança*

Estes arquivos de extensão XHTML controlam a diagramação dos componentes na tela, por meio de *tags* e estilos HTML implementados pelo

framework PrimeFaces. A Listagem 2 apresenta parte da diagramação dos componentes na tela de listagem de *baselines*.

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
            xmlns:h="http://java.sun.com/jsf/html"
            xmlns:f="http://java.sun.com/jsf/core"
            xmlns:p="http://PrimeFaces.org/ui"
            xmlns:ui="http://java.sun.com/jsf/facelets"
            xmlns:c="http://java.sun.com/jsf/composite/comps">

   <h:form id="frmprincipal" prependId="false">
      <p:dataTable  var="baseline"
                  value="#{baselineBean.listItems}"
                  id="tabela"
                  selection="#{baselineBean.selectedItem}"
                  selectionMode="single"
                  rowKey="#{baseline.id}"
                  emptyMessage="Nenhum registro encontrado">

         <f:facet name="header">
            <div class="header-crud">
               <p:commandButton id="novoRegistro"
                           value="Nova baseline"
                           actionListener="#{baselineBean.addItem()}"
                           update=":form:formDetalhes"
                           oncomplete="PF('dlgformDetalhes').show()"
                           rendered="#{mainMenuBean.aluno}"/>
               <p:commandButton id="cmdLiberar"
                           value="Liberar"
                           update=":formLiberacao"
                           oncomplete="PF('dlgFormLiberacao').show()"
                           disabled="#{empty baselineBean.selectedItem or
baselineBean.selectedItem.status == 'LIBERADA'}"
                           rendered="#{mainMenuBean.aluno}"/>
               <p:commandButton id="cmdExcluir"
                           value="Excluir"
                           actionListener="#{baselineBean.delete}"
                           update="tabela"
                           disabled="#{empty baselineBean.selectedItem or
baselineBean.selectedItem.status == 'LIBERADA'}"
                           rendered="#{mainMenuBean.aluno}">
                  <p:confirm header="Confirmação" message="Confirma a exclusão?" />
               </p:commandButton>
               <p:commandButton id="view"
                           value="Visualizar"
                           update=":baselineView"
                           oncomplete="PF('dlgbaselineViewDlg').show()"
                           disabled="#{empty baselineBean.selectedItem}"/>

               <p:commandLink id="atualizar"
                           value="Atualizar"
                           update="tabela"
                           style="float: right"/>
```

```
            </div>
        </f:facet>

        <p:ajax event="rowSelect"   update="novoRegistro cmdLiberar view cmdExcluir" />
        <p:ajax event="rowUnselect" update="novoRegistro cmdLiberar view cmdExcluir"/>

        <f:facet name="header">Lista de Baselines Cadastradas</f:facet>

        <p:column headerText="Nome" width="100px" style="text-align: center">
            <h:outputText value="#{baseline.nome}" />
        </p:column>
        <p:column headerText="Módulo" width="100px" style="text-align: center">
            <h:outputText value="#{baseline.moduloProjeto.nome}" />
        </p:column>
```

Listagem 2 – Conteúdo XHTML da página de listagem de *baselines*

Estas páginas fazem referência a atributos e/ou métodos Java devidamente mapeados por uma classe denominada *Bean*. Estes *Beans* equivalem à camada de negócio que contém os tratamentos lógicos do sistema, como mostra a Listagem 3.

```
@Named
@ViewScoped
public class BaselineBean extends GenericBean<Baseline> {

    private static final long serialVersionUID = 1L;

    private Projeto projetoSelecionado;
    private ModuloProjeto moduloSelecionado;
    private ItemDeConfiguracao itemSelecionado;
    private String tipoItemSelecionado;
    private GestaoConfiguracaoUtil gestaoConfiguracaoHelper;

    public BaselineBean() {
    }

    @Inject
    public BaselineBean(BaselineDao dao) {
        super(dao);
        projetoSelecionado = new Projeto();
        moduloSelecionado = new ModuloProjeto();
        itemSelecionado = new ItemDeConfiguracao();
        gestaoConfiguracaoHelper = new GestaoConfiguracaoUtil();
    }

    @Override
    public void save() {
        if (!this.getSelectedItem().getItensDeConfiguracao().isEmpty()) {
            super.getSelectedItem().setStatus(StatusBaseline.AGUARDANDO_LIBERACAO);
            super.getSelectedItem().setDataHoraCriacao(new Date());
            super.getSelectedItem().setUsuarioCriador(getUsuarioLogado());
            super.getSelectedItem().setModuloProjeto(this.moduloSelecionado);
            super.save();
```

```
        } else {
            FacesContext.getCurrentInstance().addMessage(null,
                    new FacesMessage(FacesMessage.SEVERITY_WARN, "Problema ao tentar salvar a
baseline",
                            "É ogrigatória a inserção de ao menos um item de configuração\n"));
            RequestContext.getCurrentInstance().update("growl");
        }
    }
```

Listagem 3 – Classe com atributos e métodos inerentes às *baselines*

Por fim os *Beans* acessam métodos e/ou atributos de classes responsáveis pela manipulação e persistência no banco de dados, as classes *Data Access Objects* (DAO). Estes objetos são separados em interface e implementação, para organização das informações e também melhor aplicação das heranças Java. A Listagem 4 mostra a interface DAO para o cadastro de *baselines* e a classe genérica DAO que define os métodos a serem implementados por esta interface.

```
package br.edu.utfpr.projetotcc.dao.interfaces;


import br.edu.utfpr.projetotcc.dao.generic.GenericDAO;
import br.edu.utfpr.projetotcc.model.Baseline;

public interface BaselineDao extends GenericDAO<Baseline>{

}


package br.edu.utfpr.projetotcc.dao.generic;

import java.util.List;

public interface GenericDAO<T>{

    void beginTransaction();

    void commitTransaction();

    void save(T object);

    void delete(T object);

    List<T> listAll();

    T findById(Object id);

}
```

Listagem 4 – Interfaces Java para implementação dos DAOs

A Listagem 5 mostra a definição HTML do componente utilizado para a construção da matriz de impacto de um artefato. O componente é uma árvore com distribuição horizontal, onde os nós são montados no Java, como mostra a Lista 6. O controle dos nós retráteis é feito implicitamente pelo *JavaScript* do *PrimeFaces*.

```
<h:form id="formMatriz" style="width: 100%;">
        <p:layout style="height: 470px; width: 700px">
          <p:layoutUnit position="north" size="270" header="Impacto Artefato">
            <p:tree value="#{itemDeConfiguracaoBean.matriz}" var="node"
orientation="horizontal">
              <p:treeNode>
                <h:outputText value="#{node}" />
              </p:treeNode>
            </p:tree>
          </p:layoutUnit>
        </p:layout>
</h:form>
```

Listagem 5 - Definição html do componente utilizado na matriz de impacto

A Listagem 6 apresenta como funciona a montagem da árvore retrátil utilizada na matriz de impacto de um item de configuração. Os nós desta árvore são do tipo *TreeNode* e possuem um método de criação que recebe dois parâmetros. O primeiro parâmetro é o nome do nó, o qual será exibido na tela, o segundo parâmetro indica qual a raiz deste nó. *Quando informado null* para a raiz, então o nó torna-se a própria raiz. No exemplo abaixo a raiz recebe o nome do artefato, em seguida o segundo nível de nós é composto pelos nós de casos de uso, o terceiro nível (de tipo de artefato) é adicionado aos nós do segundo nível e por fim os nós do quarto nível (artefato em si) são adicionados aos nós do terceiro nível. Não há necessidade de adicionar o terceiro e quarto nível à raiz porque a instanciação de um nó é feita passando a referência de um objeto, portanto o nó da raiz sempre conterá os nós filhos atualizados.

```
public TreeNode montarMatrizImpacto() {
      TreeNode raiz = new DefaultTreeNode(getNomeArtefato(item), null);
      montarNoArtefatosImpactados(raiz);

      return raiz;
}

   public void montarNoArtefatosImpactados(TreeNode raiz) {
        for (CasoDeUso casoDeUso : casosDeUsoImpactados) {

            TreeNode noCasoDeUso = new DefaultTreeNode(casoDeUso.getNome(), raiz);

            List<Artefato> artefatos =
```

```
getArtefatosFromCasosDeUso(casoDeUsoEArtefatoDao.findByIdCasoDeUso(casoDeUso.getIdCaso
DeUso()));

            if (!artefatos.isEmpty()) {
                TreeNode noPai = new DefaultTreeNode("Artefatos", noCasoDeUso);

                for (Artefato artefato : artefatos) {
                    TreeNode noFilho = new DefaultTreeNode(artefato.getNome(), noPai);
                }
            }
        }
    }
```

Listagem 6 – Código *Java* para montar o componente da árvore de impacto

4.5 DISCUSSÕES

Foi desenvolvido um aplicativo para a plataforma *Web*, sendo assim, ele provê a facilidade de manutenção característica do modelo cliente/servidor. Um dos maiores focos no desenvolvimento foi prover fácil usabilidade, portanto as interfaces foram desenvolvidas com foco nos usuários, utilizando-se da biblioteca de componentes visuais *PrimeFaces*. Por esta biblioteca implementar o conceito de *ManagedBeans*, associando objetos HTML aos objetos Java, a utilização do MVC - o que tornou a arquitetura do projeto consistente e de fácil manutenção - foi realizada de forma eficaz e prática.

O uso do *My*SQL *Workbench* permitiu a criação do banco de dados do sistema sem a manipulação direta de expressões *Structured Query Language* (SQL), tornando mais fácil seu desenvolvimento, bastando criar a modelagem e utilizar de técnicas de engenharia reversa, providas pela própria ferramenta, para gerar as expressões SQL necessárias para a criação do banco.

O uso do Hibernate permitiu a associação direta de objetos Java às tabelas do banco de dados, bastando apenas criar a classe de entidade e mapeá-la no arquivo de configuração do Hibernate para que se possa fazer operações na tabela do banco. Isto simplificou de forma extrema a manipulação dos dados, uma vez que o *Java Persistence* API (JPA) e *Java Transaction* API (JTA) implementam todo o baixo nível das persistências e transações, respectivamente.

Destaca-se, ainda, o gerenciamento realizado pelo *Maven*, que compila e implanta o aplicativo sem que o programador se preocupe em baixar os pacotes Java dos quais o projeto depende.

Dentre as principais dificuldades encontradas durante o desenvolvimento do projeto, pode-se afirmar que o fato de usar um servidor externo à IDE *(GlassFish* com *NetBeans)* diminuiu consideravelmente a produtividade, pois, técnicas de depuração de código tornam-se muito mais difíceis de serem aplicadas.

Ao trabalhar com a plataforma Web é importante lembrar-se de que os navegadores armazenam cada vez mais informações em cache, portanto, muitas vezes quando altera-se um código *JavaScript*, HTML ou CSS na aplicação, o *browser* pode não reconhecer na primeira execução, por estar pegando justamente

dados da *cache*.Para resolver este problema basta atualizar a página utilizando o comando CTRL+F5, ou então limpar a *cache* manualmente.

Outra ressalva importante identificada neste projeto é na questão de *performance* quando se trabalha com tabelas com múltiplos relacionamentos mapeados por JPA. Muitas vezes durante o desenvolvimento foi preciso listar entidades com múltiplos relacionamentos e à medida que a base de dados crescia, a demora na varredura dos dados crescia proporcionalmente. Foi identificado então que o JPA, ao carregar uma entidade do banco, carregava também todos os dados das entidades relacionadas com esta, mesmo quando esses não eram utilizados após a pesquisa. Para resolver este problema e incrementar a *performance* foi utilizada a *tag Fetch.LAZY*, como mostrado na Listagem 7, para trazer os dados de entidades secundárias apenas quando estes forem requisitados pelo usuário e não quando a entidade que dá origem ao relacionamento é requisitada.

```
@Entity
@Table(name = "artefatos")
@NamedQueries({
   @NamedQuery(name = "Artefato.findAll", query = "SELECT a FROM Artefato a"),
   @NamedQuery(name = "Artefato.findByIdArtefatos", query = "SELECT a FROM Artefato a
WHERE a.idArtefatos = :idArtefatos"),
   @NamedQuery(name = "Artefato.findByNome", query = "SELECT a FROM Artefato a WHERE
a.nome = :nome"),
   @NamedQuery(name = "Artefato.findByDescricao", query = "SELECT a FROM Artefato a
WHERE a.descricao = :descricao"),
   @NamedQuery(name = "Artefato.findByModulo", query = "SELECT a FROM Artefato a
WHERE a.moduloProjeto.moduloProjetoPK = :moduloProjetoPK")})
public class Artefato implements Serializable {
   private static final long serialVersionUID = 1L;

   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   @Basic(optional = false)
   @Column(name = "idArtefatos")
   private Integer idArtefatos;

   @Basic(optional = false)
   @NotNull
   @Size(min = 1, max = 60)
   @Column(name = "nome")
   private String nome;

   @Basic(optional = false)
   @NotNull
   @Size(min = 1, max = 500)
   @Column(name = "descricao")
   private String descricao;
```

```
@Basic(optional = false)
@NotNull
@Lob
@Column(name = "artefato")
private byte[] artefato;

@OneToMany(cascade = CascadeType.ALL, mappedBy = "artefato", fetch =  FetchType.LAZY)
private List<CasoDeUso_e_Artefato> casoDeUsoEArtefatoList;
```

Listagem 7 – Entidade que utiliza carregamento lento ou por demanda

5 CONCLUSÃO

O principal objetivo deste trabalho foi compreender o processo de gestão de configuração em projetos de software, conforme o modelo de qualidade MPS.BR, a fim de especificar os requisitos necessários, realizar a modelagem e implementação de um sistema que servirá como complemento ao sistema de gestão de projetos já existente para a disciplina de Oficina de Projeto e Desenvolvimento de Software.

Desta forma as necessidades previstas no escopo deste projeto para aplicação desenvolvida foram sanadas. Apesar do sistema gerado neste trabalho ser um complemento a um software de gerenciamento de projetos já existente, ele pode ser aplicado como gestor de configuração em projetos de softwares isolados, sendo eficiente no atendimento aos resultados desse processo exigidos pelo nível F do MPS.BR em empresas de qualquer porte.

Durante a construção do sistema foi identificada a possibilidade de futuramente alterar a forma como artefatos de um projeto de software são vinculados entre si. Atualmente o vínculo se baseia em um caso de uso, portanto, se dois ou mais artefatos estão vinculados a um caso de uso comum, eles estão indiretamente vinculados. A abordagem proposta para futuras revisões e complementações do sistema, é permitir que no cadastro de um artefato o usuário possa informar os relacionamentos com outros artefatos do projeto, sendo assim, quando um deles sofrer uma alteração o sistema irá recuperar apenas artefatos (itens de configuração) impactados diretamente pela mudança, facilitando o trabalho do gerente de configuração ou gerente de projeto. Outra sugestão de melhoria identificada é com relação à necessidade de mudança de versão de um item de configuração gerada a partir do impacto gerado por uma alteração, o que atualmente é realizada manualmente pelo gerente de configuração ou de projeto a partir do histórico de artefatos. Desta forma, a intenção futura é permitir a mudança de versão seguida da matriz de impacto gerada logo após a confirmação de uma mudança em um item de configuração, evitando a busca do artefato em seu histórico para proceder a mudança de versão.

Destaca-se a importância desse trabalho para o acadêmico e para futuros acadêmicos e profissionais da área, contribuindo para o aprendizado de tecnologias

atuais aplicadas ao desenvolvimento de sistemas Web, bem como para o aprendizado de um processo de grande importância a fim de manter a consistência de projetos de software a fim de gerar produtos de qualidade em seu processo de desenvolvimento.

REFERÊNCIAS

ASKLUND, U., BENDIX, L., A study of configuration management in open source software, **IEE Proceedings - Software**, v. 149, n. 1, p. 40-46, Fev. 2002.

ATLASSIAN, **JIRA**, Disponível em: <http://www.atlassian.com/software/jira>. Acesso em: 28 ago. 2014.

BEZERRA, Eduardo. **Princípio de análise e projeto de sistemas com UML**, Elsevier - Campus, 2006.

GIT, **About,** Disponível em: <http://www.git-scm.com/about>. Acesso em: 03 set. 2014.

IEEE, **Std 610.12 - IEEE Standard glossary of software engineering Terminology**, Institute of Electrical and Electronics Engineers, 1990.

IEEE, **Std 828 - IEEE Standard for software configuration management plans,** Institute of Electrical and Electronics Engineers, 2005.

MURTA, L. G. P. **Gerência de configuração no desenvolvimento baseado em componentes**. Tese (Doutorado em Ciências em Engenharia de Sistemas de Computação) – Programa de Pós-graduação em Engenharia pela Universidade Federal do Rio de Janeiro, 2006.

PEREIRA, Anna Carolina. **Implantando gerência de configuração com apoio da ferramenta Subversion (SVN) para atingir os resultados esperados do processo GCO do nível F do MPS.BR**. Tese (Pós-Graduação em Melhoria de Processo de Software) – Programa de Pós-graduação em Melhoria de Processo de Software pela Universidade Federal de Lavras, 2008.

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

SOFTEX. **Guia de Implementação Parte 9: Implementação do MR-MPS em Organizações do tipo Fábrica de Software.** Novembro 2011.

SOMMERVILLE, Ian. **Engenharia de software**. 6. ed. Tradução Maurício de Andrade. São Paulo: Addison Wesley, 2003.

SHRUM, Sandi. **Continuous and staged, a choice of CMMI representations**. Disponível em: <http://www.sei.cmu.edu/library/abstracts/news-at-sei/spotlightdec99.cfm>. Acesso em: 21 set. 2014.