SISTEMA DE GESTÃO DE PROJETOS DE SOFTWARE - SGPS

Luciene Chagas de Oliveira, Universidade de Uberaba (UNIUBE) — E-mail: luciene.oliveira@uniube.br
Mateus Silvestre da Silva, Universidade de Uberaba (UNIUBE) — E-mail: mateus.silvestre.silva@gmail.com
Mylene Lemos Rodrigues, Universidade de Uberaba (UNIUBE) — E-mail: mylene.rodrigues@uniube.br
Lilian Ribeiro Mendes Paiva, Universidade de Uberaba (UNIUBE) — E-mail: lilian.paiva@uniube.br
Mariana Davi Justino, Universidade de Uberaba (UNIUBE) — E-mail: marianadavij@hotmail.com

RESUMO: A Engenharia de Software é uma área do conhecimento da computação muito importante para a produção de software, que contempla desde a especificação, desenvolvimento, testes até a manutenção de sistemas aplicando tecnologias e práticas de gerência de projetos e outras disciplinas, o que objetiva na organização, produtividade e qualidade. O processo de construção de software necessita de um constante gerenciamento de projeto de software. Em virtude disso, surge a necessidade de ferramentas para facilitar a gestão de projeto de software. O objetivo deste trabalho é propor um modelo de Sistema de Gestão de Projetos de Software (SGPS) que visa realizar o controle das etapas do desenvolvimento de software.

Palavras-chave: gestão de projetos; engenharia de software; sistemas de informação

SYSTEM OF SOFTWARE PROJECT MANAGEMENT

Abstract: Software Engineer is a very important area to software production, which owns since specification, development and test until system maintenance, applying project management technologies and approaches among other contents. Software construction process needs a constant software project management. Indeed there is the need to create tools to make project management easier. The goal of this paper is to propose a Project Management System, called SGPS that aims to control all software development steps.

Keywords: project management; software engineering; information systems

1. INTRODUÇÃO

A Engenharia de Software é uma área de extrema importância no mundo da computação, relacionada com todos os aspectos da produção de software, desde os estágios iniciais de especificação do sistema até sua manutenção, depois que este entrar em operação (SOMMERVILLE, 2006).

Um processo de software é um conjunto de atividades que levam à produção de software. Embora existam muitos processos de softwares diferentes, algumas atividades são comuns a todos:

1. Especificação de software: definição das funcionalidades do software e das restrições sobre sua operação.

2. Projeto (Arquitetura) e Implementação de software: processo de conversão da especificação do sistema em um sistema executável.

3. Validação de software: o software deve ser validado para garantir as necessidades do cliente.

4. Evolução de software: atividade de manutenção de software.

Atualmente, a Gestão de Projetos, destaca-se entre as áreas da engenharia de software, crescendo de forma continua nas empresas. Consiste em metodologias, técnicas, conhecimentos técnicos, habilidades e ferramentas na condução das atividades da produção de um projeto de software visando atender os seus objetivos (PMI, 2004).

Diante das necessidades na área de Engenharia de Software e Gestão de Projetos, o desenvolvimento de um software para realizar esta gestão faz-se necessário. Portanto, o objetivo deste artigo é apresentar o modelo de um sistema de Gestão de Projetos de Software

(SGPS) para auxiliar no controle de todas as fases de desenvolvimento do mesmo: Especificação de Requisitos, Projeto, Desenvolvimento, Testes e Implantação.

Este sistema permitirá também realizar atividades de gestão de projetos como Cronograma, Custos, Gestão de Riscos e Gestão de Mudanças. O SGPS poderá ser utilizado por toda equipe de desenvolvimento de um software: Gerentes de Projetos, Analista de Requisitos, Arquitetos de Software, Testadores e Suportes de sistemas de informação. Sendo assim, como o gerenciamento de projetos de software é uma parte essencial da engenharia de software, este sistema irá auxiliar em toda a construção e gestão de projetos de software nas empresas de desenvolvimento de software ou em projetos de software em instituições de ensino.

O sistema de gestão de projetos possui os seguintes módulos: login do usuário, controle de acesso, módulo de gestão de requisitos, módulo de gestão da arquitetura (projeto de software), módulo de estimativa (cronograma), gestão de prazos, custos e atividades, módulo de Workflow (fluxo de aprovação), módulo de gestão de mudanças, módulo de gestão de testes, módulo de gestão de implantação e módulo de gestão de riscos.

O grande diferencial deste sistema está nos benefícios das áreas de gestão de um projeto de software: gestão de projetos, gestão de requisitos, gestão do tempo, gestão do custo, gestão de mudanças, gestão de riscos e gestão de implantação. A maioria dos sistemas existentes não envolvem as áreas de gestão contempladas no SGPS. Outro diferencial é que o SGPS proporciona aos seus usuários a maioria dos módulos de gestão da engenharia de software, tornado o produto capaz de solucionar os processos para o desenvolvimento de um software.

2. Gestão de projetos

De acordo com (SOTILLE, 2004), o gerenciamento de projetos de software é uma parte essencial da engenharia de software, composto por algumas das seguintes áreas de gestão:

1. Gestão de Tempo: é necessário elaborar um cronograma, identificando os marcos, atividades e recursos para o projeto.
2. Gestão de Custos: o gerenciamento de orçamento é necessário para atender as metas da empresa que está desenvolvendo o software.
3. Gestão de Atividades: é muito importante gerenciar as atividades realizadas pelos recursos envolvidos em um projeto de software, bem como a produtividade das pessoas.
4. Gestão de Mudanças: as mudanças são inevitáveis. Assim que o software é colocado em uso, novos requisitos surgem e os requisitos existentes são alterados à medida que a empresa passa por modificações. Devido às mudanças ocorridas, é importante realizar o gerenciamento de mudanças.
5. Gestão de Riscos: área que identifica, qualifica, controla, quantifica os riscos do projeto. O gerenciamento de riscos está sendo considerado, cada vez mais, como uma das principais atividades dos gerentes de projeto.

A gerência de projetos é a primeira camada do processo de engenharia de software. Ela abrange todo o processo de desenvolvimento de software, do começo ao fim.

A linguagem de modelagem UML

A UML (Unified Modeling Language) tornou-se, atualmente, a linguagem padrão de modelagem adotada internacionalmente pela indústria de engenharia de software. É uma linguagem visual utilizada para modelar softwares baseados no paradigma de orientação a

objetos, cujo propósito geral que se pode ser aplicado a todos os domínios de aplicação (GUEDES, 2009).

Com a modelagem, são alcançados quatro objetivos (BOOCH, RUMBAUGH, JACOBSON, 2006):

1. Os modelos ajudam a visualizar o sistema como ele é ou como deseja-se que seja. Um modelo é uma simplificação da realidade.
2. Os modelos permitem especificar a estrutura ou o comportamento de um sistema.
3. Os modelos proporcionam um guia para a construção do sistema.
4. Os modelos documentam as decisões tomadas.

Nas fases de concepção e elaboração do sistema foi elaborada uma documentação de requisitos e arquitetura contendo modelos de software utilizando a linguagem UML.

A tecnologia Java

Para a fase de construção do sistema são utilizados os conceitos do paradigma de desenvolvimento orientado a objetos, bem como, desenvolvimento web utilizando a linguagem Java, plataforma JEE (Java Enterprise Edition), Banco de Dados e frameworks de desenvolvimento, JSF (Java Server Faces), Hibernate, Spring, entre outros.

A Orientação a Objetos é um paradigma de análise, projeto e programação de sistemas baseado na composição e interação entre diversas unidades de software. É uma forma especial de programar, mais próximo de como são expressos os objetos na vida real (DEITEL et. al. 2006).

A linguagem Java, que será utilizada neste projeto, é uma poderosa linguagem de programação orientada a objetos para desenvolvimento de aplicações Web, Desktop e para dispositivos móveis (CAMPIONE, 1996). Atualmente, cresce cada vez mais o número de desenvolvedores que desejam escrever aplicativos transacionais distribuídos para corporação e desse modo alavancar a velocidade, a segurança e a confiabilidade da tecnologia. Para reduzir os custos e acelerar o projeto do aplicativo e do desenvolvimento, a plataforma Java Enterprise Edition (JEE) fornece uma metodologia baseada em componentes para projeto, desenvolvimento e implantação de aplicações empresariais. A plataforma JEE oferece um modelo de aplicativo distribuído multicamada, componentes reutilizáveis, um modelo de segurança unificado, controle flexível de transações e suporte a serviços Web através do intercâmbio de dados integrados em padrões abertos (STEPHANIE, 2005).

Sistema de gestão de projetos de software – SGPS

O sistema de gestão de projetos de software (SGPS) está dividido em módulos do gerenciamento de projetos e da engenharia de software, descritos a seguir.

a) Gestão de requisitos

A Gerência de Requisitos tem como objetivo principal controlar a evolução dos requisitos, seja por constatação de novas necessidades, seja por constatação de deficiências nos requisitos registrados até o momento (THAYER, DORFMAN, 1997).

Este controle é importante, pois a não identificação e compreensão dos requisitos, ou mudanças feitas durante o ciclo do projeto não controladas ou não compreendidas por todos os envolvidos são algumas das causas para o insucesso de projetos de software. Os requisitos de um sistema são descrições dos serviços fornecidos pelo sistema e as suas restrições operacionais. Esses requisitos refletem as necessidades dos clientes de um sistema que ajudam a resolver algum problema, por exemplo, controlar um dispositivo, enviar um pedido

ou encontrar informações. O processo de analisar, validar, verificar, documentar e restringir é chamado de engenharia de requisitos. O módulo de requisitos compreende em contemplar no sistema toda a especificação de requisitos de um software. A especificação de software, atualmente chamada de Engenharia de Requisitos, consiste em definir as funcionalidades do sistema, que possui as etapas: estudo de viabilidade do software, levantamento e análise de requisitos, elaboração da documentação de requisitos e validação dos requisitos do software. A gestão de requisitos tem como objetivo estabelecer uma visão comum entre o cliente e a equipe do projeto em relação aos requisitos que serão atendidos pelo software (SOMMERVILLE, 2006).

De modo geral, os requisitos podem ser classificados em: requisitos do usuário, requisitos de sistema, requisitos de domínio, requisitos funcionais, requisitos não funcionais e requisitos inversos. Requisitos do usuário são declarações, explicações, em linguagem natural e também em diagramas, sobre as funções do sistema e suas restrições operacionais, de modo que sejam compreensíveis. Os leitores deste tipo de especificação geralmente são aqueles que não estejam interessados como o sistema será implementado ou nos recursos detalhados do sistema (Ex.: usuários finais de sistemas) (SOMMERVILLE, 2006).

Requisitos do usuário são declarações, explicações, em linguagem natural e também em diagramas, sobre as funções do sistema e suas restrições operacionais, de modo que sejam compreensíveis. Os leitores deste tipo de especificação geralmente são aqueles que não estejam interessados como o sistema será implementado ou nos recursos detalhados do sistema (Ex.: usuários finais de sistemas) (SOMMERVILLE, 2006).

Os requisitos de sistema são descrições mais detalhadas dos requisitos do usuário. Eles podem servir como base para um contrato detalhado à implementação do sistema e, portanto, devem ser uma especificação completa e consistente de todo o sistema. Eles são utilizados pelos engenheiros de software como ponto de partida para o projeto de sistema. Os leitores deste tipo de especificação precisam saber bem detalhadamente o sistema (Ex.: Desenvolvedores de software, Arquitetos de sistemas) (SOMMERVILLE, 2006).

Os requisitos de domínio são derivados do domínio da aplicação do sistema, em vez de serem obtidos a partir das necessidades específicas dos usuários do sistema. Eles podem ser novos requisitos funcionais em si, podem restringir os requisitos funcionais existentes ou estabelecer como devem ser realizados cálculos específicos (SOMMERVILLE, 2006).

Os requisitos funcionais para um sistema descrevem a interação entre o sistema e o seu ambiente, ou seja, a funcionalidade ou os serviços que se espera que o sistema forneça. Eles dependem do tipo de software que está sendo desenvolvido, dos usuários de software que se espera verificar e do tipo de sistema que está sendo desenvolvido (SOMMERVILLE, 2006).

Os requisitos não funcionais descrevem qualidades, características, comportamento que o software deve ter. São aqueles que não dizem respeito diretamente às funções específicas fornecidas pelo sistema, como confiabilidade, tempo de resposta, espaço em disco e interoperabilidade (SOMMERVILLE, 2006).

Para todos os sistemas, o processo de engenharia de requisitos de sistema deve começar com um estudo de viabilidade. A entrada para o estudo da viabilidade é uma descrição geral do sistema e de como ele será utilizado dentro de um a organização (SOMMERVILLE, 2006). Os resultados do estudo de viabilidade devem ser um relatório que recomenda se vale a pena ou não realizar o processo de engenharia de requisitos e o processo de desenvolvimento de sistemas.

Outra atividade da engenharia de requisitos é a elaboração do documento de requisitos de software. O documento de requisitos de software, chamado também de SRS (*Software*

Requeriments Specification), é a declaração oficial do que os desenvolvedores de sistema devem implementar. Ele deve incluir os requisitos de usuário de um sistema e uma especificação detalhada dos requisitos do sistema. O documento de requisitos de software deve ser organizado de tal forma que os envolvidos no software possam usá-lo (SOMMER-VILLE, 2006).

Parte do processo de garantia da qualidade de software e gerência de requisitos consiste na realização periódica dos requisitos do software, bem como a inspeção dos mesmos. O processo de inspeção de requisitos caracteriza-se pela leitura cuidadosa do documento de requisitos com utilização de uma técnica de leitura, buscando a localização de erros ou defeitos no mesmo, segundo um critério pré-estabelecido. A literatura aponta que o processo de inspeção pode detectar de 30% a 90% dos erros existentes nos artefatos gerados em um processo de desenvolvimento de software (LAITENBERGER, 2001).

Na Figura 1, é apresentado o diagrama de caso de uso contendo as funcionalidades do módulo de Gestão de Requisitos. O módulo de gestão de requisitos do sistema de gestão de projetos oferece para as organizações e/ou instituições de ensino, todo o gerenciamento de requisitos desde o documento de viabilidade até a inspeção de requisitos. O módulo de gestão de requisitos é um fator crucial para estabelecer todos os requisitos que o projeto venha de ter e, além disso, promover a qualidade do software.

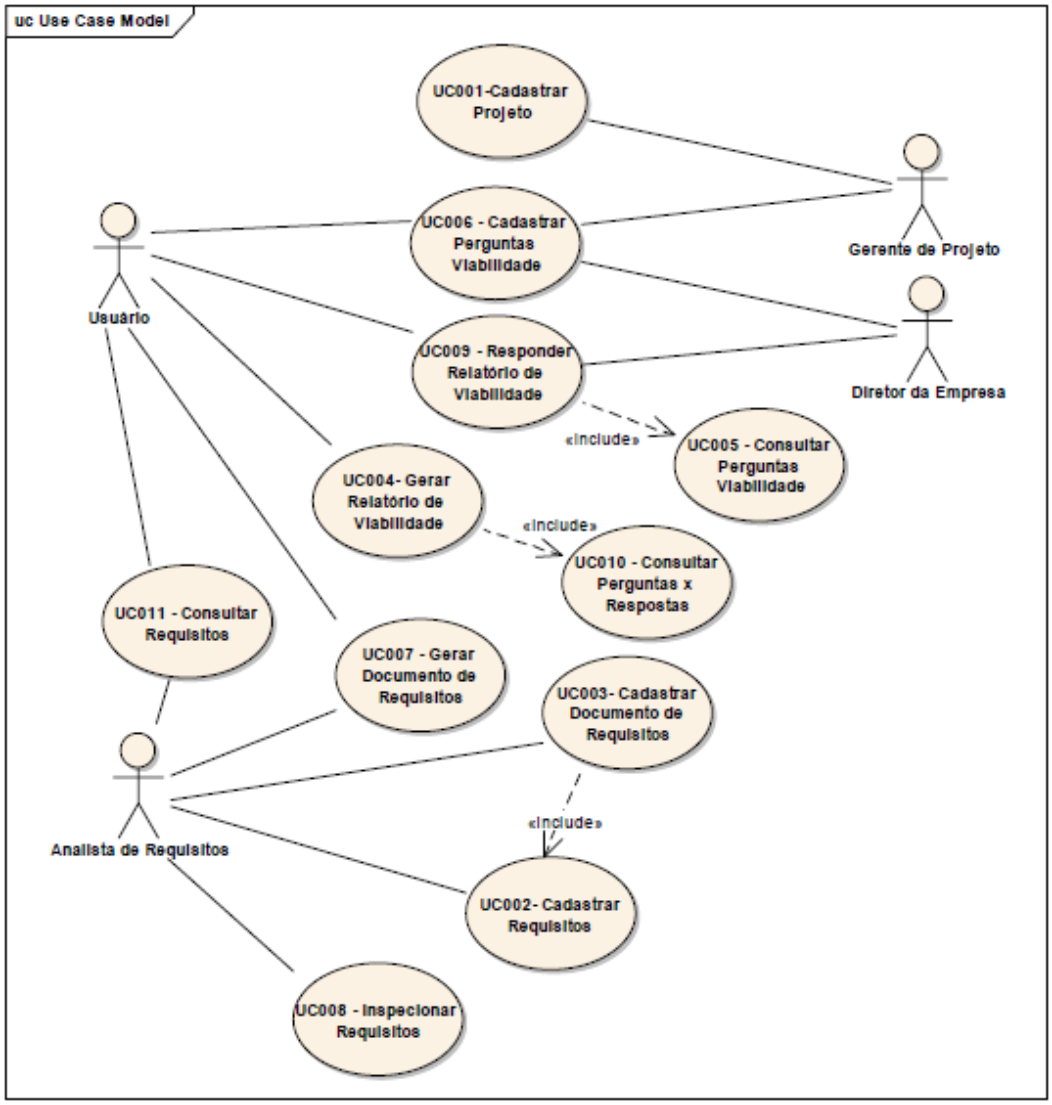

Figura 1. Casos de Uso do módulo de Gestão de Requisitos

b) Gestão de tempo

O tempo sempre foi um fator importante em projetos. Tanto é que a própria definição de projeto faz referência ao tempo, pois projeto é um "esforço temporário empreendido para criar um produto, serviço ou resultado exclusivo" (PMBOK, 2004).

O módulo de gestão de tempo compreende na elaboração do cronograma, contendo as atividades e recursos envolvidos em um projeto de software. Existem quatro aspectos da Gestão de Tempo que fazem diferença em um projeto de software: tamanho, esforço, qualidade e cronograma. O processo da gestão de tempo é composto pelas seguintes etapas:

- Definição das Atividades: identificação das atividades específicas que devem ser elaboradas para se produzir os produtos/serviços do projeto, bem como de suas várias entregas.
- Definição da Sequência das Atividades: identificação e documentação das relações de dependência entre as atividades do projeto.
- Estimativa dos Recursos das Atividades: estima os recursos necessários para a elaboração de cada atividade.
- Estimativa da Duração das Atividades: estima a quantidade de períodos de trabalho que serão necessários para a implementação de cada atividade.
- Desenvolvimento do Cronograma: analisa a sequência das atividades e respectiva duração, considerando os recursos necessários para elaborar o cronograma do projeto.
- Controle do Cronograma: controle as mudanças no cronograma do projeto.

O módulo de gestão de custos está contido no módulo de gestão de tempo, pois, de acordo com (PRESSMAN, 2004), a estimativa dos recursos, custos e a programação das atividades para um esforço de desenvolvimento de software exige experiência e acesso a boas informações históricas.

Segundo uma pesquisa realizada pelo PMI do Rio de Janeiro, no seu relatório "Estudo de Benchmarking em Gerenciamento de Projetos 2005", que mostra em determinados itens, o prazo é um fator de preocupação das empresas com relação aos projetos, conforme apresentado na Figura 2.

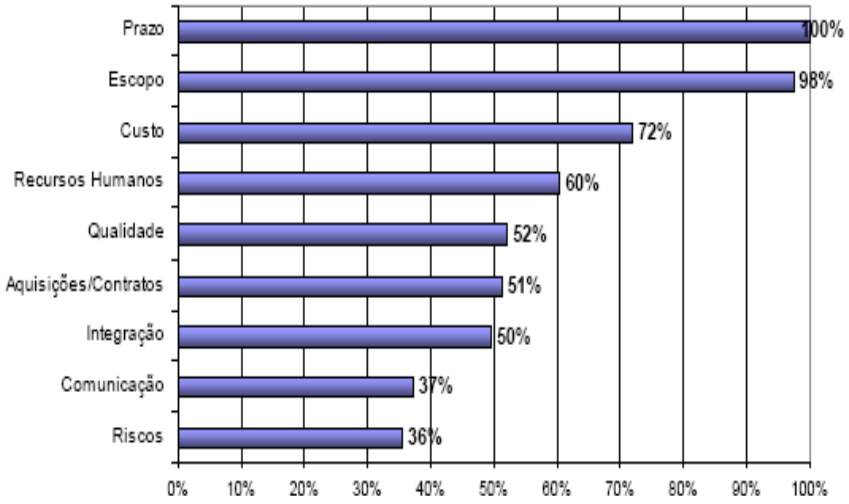

Figura 2. Aspectos mais considerados no planejamento de projetos sob uma perspectiva geral. Fonte: Project Management Institute – seção Rio de Janeiro – Estudo de Benchmarking em Gerenciamento de Projetos 2005.

Na Figura 3, é apresentado o diagrama de caso de uso contendo as funcionalidades do módulo de Gestão de Tempo.

O cronograma de um projeto de *software* envolve a definição de prazos, definição das atividades e identificação dos recursos (pessoas) envolvidos no desenvolvimento do sistema.

No processo de planejamento iterativo, o planejamento do conjunto de atividades no cronograma tem como meta a definição de uma sequência de resultados intermediários e das principais iterações. É importante que se entenda tal planejamento de forma evolucionária, uma vez que ajustes no conteúdo e no cronograma serão necessários ao longo do tempo (GARCIA, 2010).

A definição das atividades do cronograma envolve identificar e documentar o trabalho planejado para ser realizado.

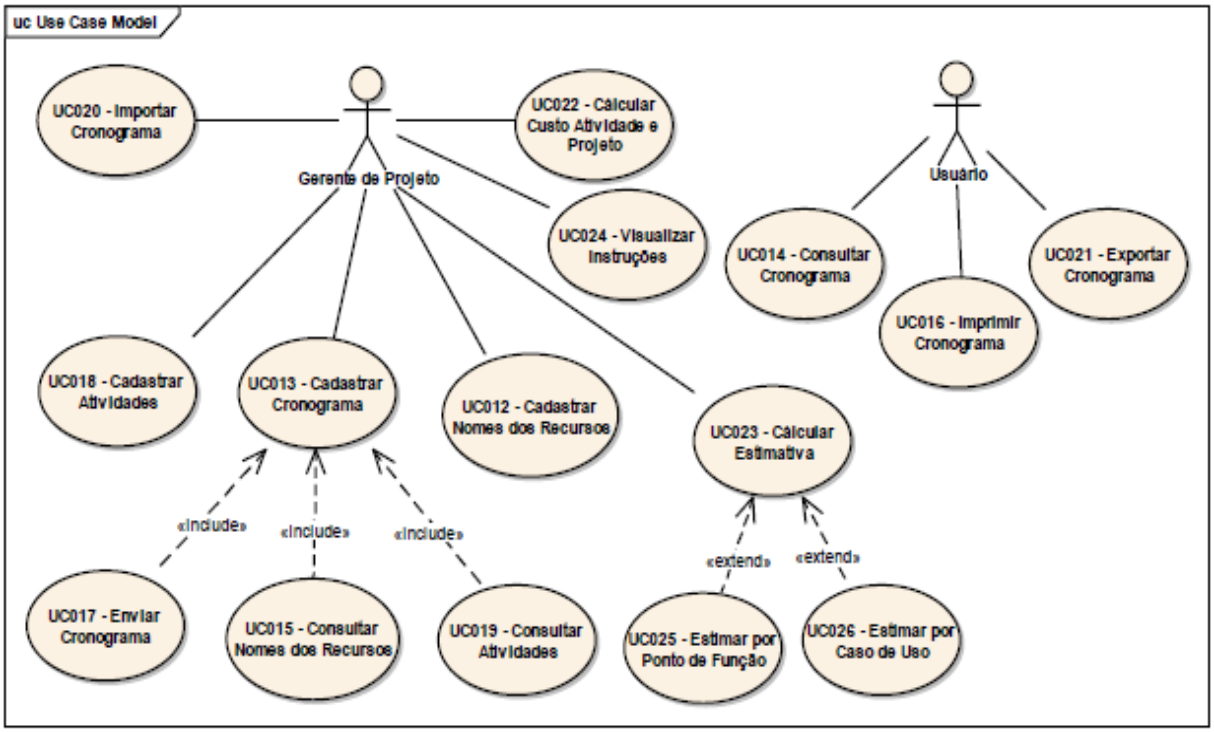

Figura 3. Casos de Uso do módulo de Gestão de Tempo

As medidas de tamanho de software surgiram com o objetivo de estimar o esforço (número de pessoas-hora) e o prazo associados ao desenvolvimento dos programas e sistemas. (AGUIAR, 2010).

As métricas de estimativas de tempo do sistema são: Pontos de Função (PF) e Pontos por Caso de Uso (UCP). Os pontos de função (PF), originalmente concebidos por Albrecht, ganharam crescente popularidade em 1986. Em 2002 os PF passaram à condição de padrão internacional, através da norma ISSO/IEC 20926. Os Pontos de Função podem ser facilmente contados ou estimados a partir de casos de uso. Diversas organizações têm utilizado esse método com sucesso. Os pontos por caso de uso (UCP) foram criados em 1993, como adaptação específica dos pontos de função. Esta métrica permite fazer estimativas no início do projeto com base no modelo de casos de uso. (AGUIAR, 2010).

A gerência do custo do projeto agrega os processos que envolvem planejamento, estimativa, orçamento e controle de custos que serão necessários para a conclusão do projeto a partir de uma previsão orçamentária.

A expectativa tanto por parte do cliente quanto por parte da gerência para que o projeto termine dentro do prazo desejado é um dos grandes desafios para as empresas de software. Portanto, a gestão do tempo é um fator decisivo para controlar as atividades do cronograma dentro do prazo e os custos afim de terminar o projeto com sucesso.

O módulo de gestão de tempo do sistema de gestão de projetos contempla para as empresas e/ou instituições de ensino, todos os processos necessários para o desenvolvimento do cronograma, as técnicas de estimativa do tempo e do custo do projeto.

c) Gestão de arquitetura

O módulo de gestão de arquitetura de projeto de software (Figura 4), compreende a inclusão de diagramas, modelos e artefatos pertencentes a fase de projeto de um software. Neste módulo, os requisitos são traduzidos em uma representação de software.

A arquitetura de software é considerada uma ferramenta para lidar com a complexidade do software. Enfatiza-se que a arquitetura deve satisfazer os requisitos funcionais e não funcionais do sistema, incrementando a definição de que arquitetura de software é o conjunto de componentes e seus relacionamentos. Portanto, é possível notar que a arquitetura de software é mais do que a descrição dos componentes que a compõem e do relacionamento entre eles. A arquitetura é a interface entre duas partes distintas: o problema de negócio e a solução técnica (VAROTO, 2002).

Similar a esta definição, Bass *et al.* (2003) diz que arquitetura de software são as estruturas que incluem componentes, suas propriedades externas e os relacionamentos entre eles, constituindo uma abstração do sistema. Esta abstração suprime detalhes de componentes que não afetam a forma como eles são usados ou como eles usam outros componentes, auxiliando o gerenciamento da complexidade.

A arquitetura de software também chamada de projeto de software é a parte da engenharia de software que se encarrega de transformar os resultados da Análise de Requisitos em um documento ou conjunto de documentos capazes de serem interpretados diretamente pelo programador (LARMAN, 2007).

Este módulo representa uma etapa crucial para o sucesso no desenvolvimento de qualquer sistema, pois com ele o projetista tem uma visão ampla do que deve ser feito e aplica a estratégia que melhor atende às necessidades do software.

A estratégia de arquitetura de software deve considerar os seguintes aspectos na concepção de um software:

- Extensibilidade: o software suporta a adesão de novas funcionalidades sem que haja necessidade de grandes alterações na sua arquitetura subjacente;
- Robustez: o software deve estar preparado para tratar de situações imprevisíveis, como entrada de dados inválida e condições de baixa memória do computador;
- Tolerância à falhas: o software não pode estar suscetível à falhas, ele deve ser resistente e capaz de recuperar possíveis ações de falha;
- Compatibilidade: o software deve ter a capacidade de operar com outros produtos que também foram projetados para suportar a interoperabilidade;
- Modularidade: o software resulta em componentes independentes e bem definidos, o que leva a uma melhor divisão de trabalho na equipe de desenvolvimento e melhor manutenção do sistema;

- Reusabilidade: um software reusável permite que seus componentes modularizados possam ser reutilizados em casos que existam necessidades semelhantes em outros projetos.

Uma arquitetura de software deve seguir padrões de projeto de sofware. De acordo com Bass, Clements e Kazman (2003), o conceito de padrão de projeto de software tem as seguintes características:

- Encapsulamento: um padrão encapsula um problema/solução bem definido. Ele deve ser independente, específico e formulado de maneira a ficar claro onde ele se aplica.
- Generalidade: todo padrão deve permitir a construção de outras realizações a partir deste padrão.
- Equilíbrio: quando um padrão é utilizado em uma aplicação, o equilíbrio dá a razão, relacionada com cada uma das restrições envolvidas, para cada passo do projeto. Uma análise racional que envolva uma abstração de dados empíricos, uma observação da aplicação de padrões em artefatos tradicionais, uma série convincente de exemplos e uma análise de soluções ruins ou fracassadas pode ser a forma de encontrar este equilíbrio.
- Abstração: os padrões representam abstrações da experiência empírica ou do conhecimento cotidiano.
- Abertura: um padrão deve permitir a sua extensão para níveis mais baixos de detalhe.
- Combinatoriedade: os padrões são relacionados hierarquicamente. Padrões de alto nível podem ser compostos ou relacionados com padrões que endereçam problemas de nível mais baixo.

A necessidade de várias visões em arquitetura de software de abstração na modelagem dos requisitos que serão implementados era percebida pelos projetistas, mas não registradas até então.

Com esta formalização, surge também o papel do arquiteto de software (VAROTO, 2002). Algumas possíveis visões são:

- Visão de Casos de Uso, que contém casos de uso e cenários que abrangem comportamentos significativos em termos de arquitetura, classes ou riscos técnicos. Ela é um subconjunto do modelo de casos de uso.
- Visão Lógica, que contém as classes de design mais importantes e sua organização em pacotes e subsistemas, e a organização desses pacotes e subsistemas em camadas. Ela contém algumas realizações de caso de uso. É um subconjunto do modelo de design.
- Visão de Implementação, que contém uma visão geral do modelo de implementação e sua organização em termos de módulos em pacotes e camadas. A alocação de pacotes e classes (da Visão Lógica) nos pacotes e módulos da Visão de Implementação também é descrita. Ela é um subconjunto do modelo de implementação.
- Visão de Processos, que contém a descrição das tarefas (processo e threads) envolvidas, suas interações e configurações, e a alocação dos objetos e classes de design em tarefas. Essa visão só precisará ser usada se o sistema tiver um grau

significativo de simultaneidade. No RUP, ela é um subconjunto do modelo de design.

- Visão de Implantação, que contém a descrição dos vários nós físicos da maior parte das configurações comuns de plataforma e a alocação das tarefas (da Visão de Processos) nos nós físicos. Essa visão só precisará ser usada se o sistema estiver distribuído. Ela é um subconjunto do modelo de implantação.

Com o amadurecimento da engenharia, percebeu-se que uma boa arquitetura é um fator decisivo de sucesso para o projeto e o desenvolvimento do sistema, visto que ela desempenha o papel de uma ponte entre requisitos e código.

O módulo de gestão de arquitetura de software do sistema de gestão de projeto tem por objetivo projetar a estrutura do sistema, desde a concepção até o seu término. Este módulo é de extrema importância para promover um bom projeto de software do sistema.

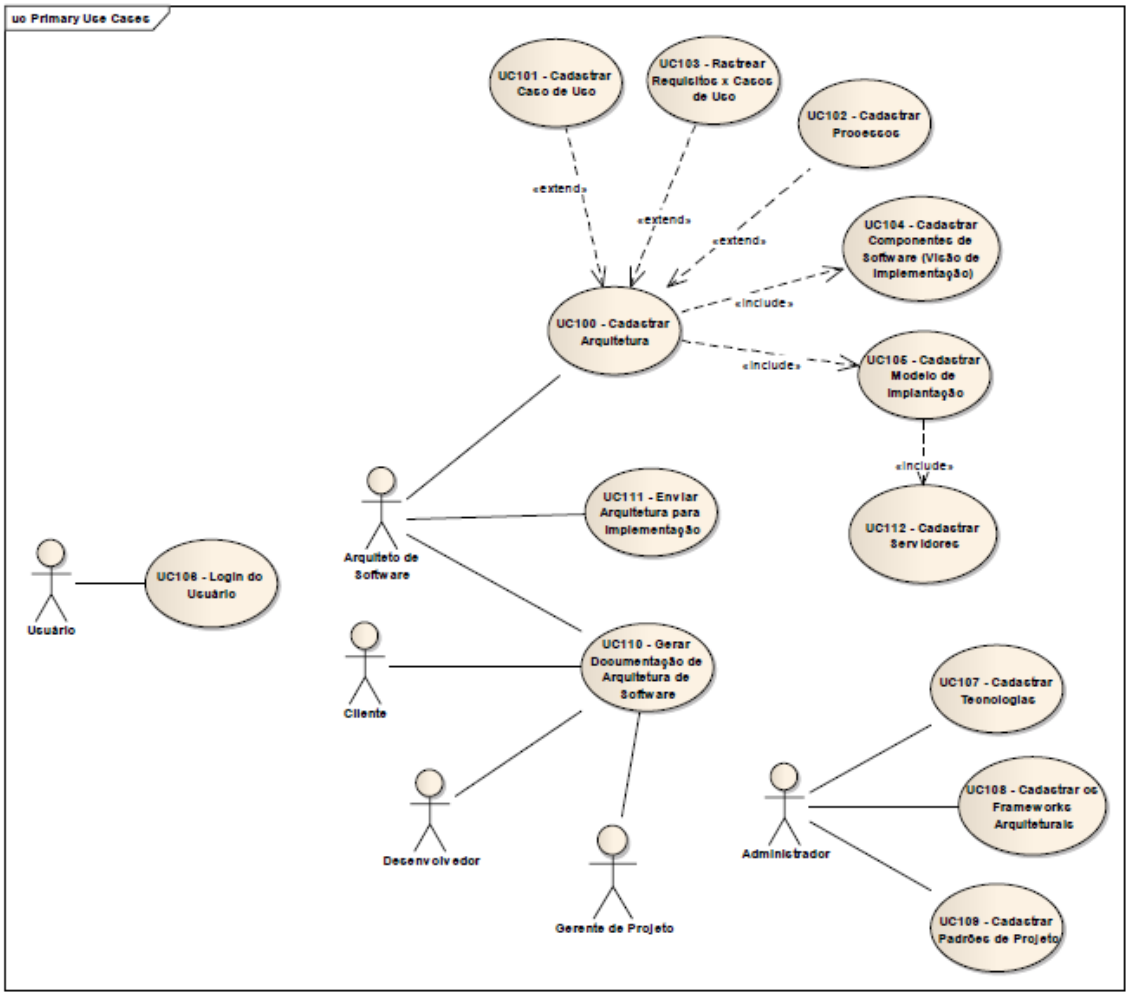

Figura 4. Casos de Uso do módulo de Gestão de Arquitetura

d) Gestão de mudanças

Outro módulo do SGPS é o de gestão de mudanças, que compreende o gerenciamento das mudanças ocorridas durante e após o desenvolvimento de um software. O gerenciamento de mudança inclui procedimentos, ferramentas e dependências que podem ser necessárias para planejar, implementar e dirigir o Gerenciamento de Mudanças, conforme pode ser visualizado no caso de uso da Figura 6. É nesta fase de Gerenciamento de Mudanças que se decide se é melhor ou não aplicar a mudança solicitada ao projeto.

Empresas de diferentes portes utilizam sistemas para gerenciar o andamento e os prazos de seus projetos. Tendo notado a escassez de sistemas que façam o gerenciamento de todas as etapas de produção de software, o SGPS criará ferramentas que permitam controle do Cronograma, Custos, Gestão de Riscos e Gestão de Mudanças.

Os sistemas computacionais são voláteis e, portanto sofrem mudanças ao logo do tempo. Para conduzir estas mudanças recomenda-se preparo e planejamento. A organização e gerenciamento das mudanças permite diferenciar o que era o requisito original, o que foi introduzido e o que foi descartado. Além disto, é interessante estabelecer um único canal para controle de mudanças, bem como utilizar um sistema para este controle.

Furlan (2001) apresenta uma sucessão de passos que devem ser seguidos para um processo de controle de mudança:

- Checar validade da solicitação de mudança;
- Identificar os requisitos diretamente afetados com a mudança;
- Identificar dependências entre requisitos para buscar os requisitos afetados indiretamente;
- Assegurar com o solicitante a mudança a ser realizada;
- Estimar custos da mudança;
- Validar com o usuário sobre o custo da mudança.

Após a realização desta análise, podemos aceitar ou rejeitar a mudança, conforme os impactos esperados no sistema. As mudanças podem ser feitas a longo ou curto prazo, dependendo das necessidades. As mudanças podem começar com a identificação de um problema, com a finalidade de solucioná-lo ou com uma proposta específica de mudança dos requisitos do sistema, porém, muitas mudanças podem vir de benefícios buscados em outras necessidades como redução de custos ou melhorias nos serviços.

Nestas condições, faz-se necessário criar um documento que estabeleça alterações feitas nos requisitos do sistema, incluindo procedimentos, ferramentas e dependências que podem ser necessárias para planejar e dirigir o Gerenciamento de Mudança. É nesta fase que se decide se é melhor ou não aplicar a mudança solicitada ao projeto.

Sommerville (2006) afirma que a vantagem de utilizar um processo formal de gerenciamento de mudanças é que todas as propostas são tratadas de modo consistente e que as mudanças no documento de requisitos são feitas de maneira controlada.Os estágios presentes no decorrer do Gerenciamento de Mudanças, apresentam-se ilustrados na Figura 5 e descritos sequencialmente:

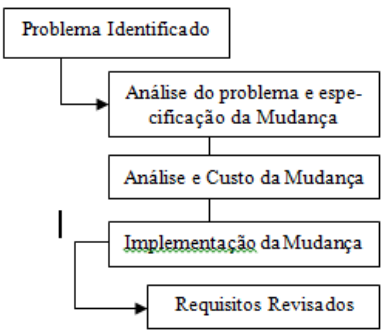

Figura 5. Estágios do gerenciamento de mudanças.

- Análise do problema e Especificação da Mudança: processo que começa com a identificação de um problema, com os requisitos ou com uma proposta de mudança específica;
- Análise e custo da Mudança: neste estagio o efeito da mudança é avaliado, utilizando informações sobre a facilidade de rastreamento e o conhecimento geral dos requisitos do sistema. O custo é estimado nas mudanças realizadas no documento de requisitos e na sua implementação;
- Implementação da mudança: o documento de requisitos deve ser organizado de maneira que futuramente as mudanças possam ser feitas sem grande retrabalho. Toda mudança, sendo fácil ou difícil deve ser colocada no documento de requisitos e, mesmo se for urgente, antes de fazê-la, deve-se alterar no documento, pois, posteriormente elas podem ser esquecidas, causando problemas futuros (PFLEEGER, 2004).

Ressalta-se que o gerenciamento das mudanças quando devidamente encarado como uma oportunidade pode passar a ser um instrumento importante no controle e na melhoria continua em direção a satisfação do usuário e aos objetivos propostos pelo projeto.

O Diagrama de Caso de Uso para Gerenciamento de Mudanças tem como atores o Gerente de Mudanças, responsável que analisa e comanda toda a parte do gerenciamento de mudanças; o Usuário, restrito a algumas funcionalidades; e o Desenvolvedor, que acessa somente a parte de mudanças aprovadas. Os mesmos estão ilustrados na figura 6, Diagrama de Casos de Uso - funcionalidades do módulo de gestão de mudanças. .

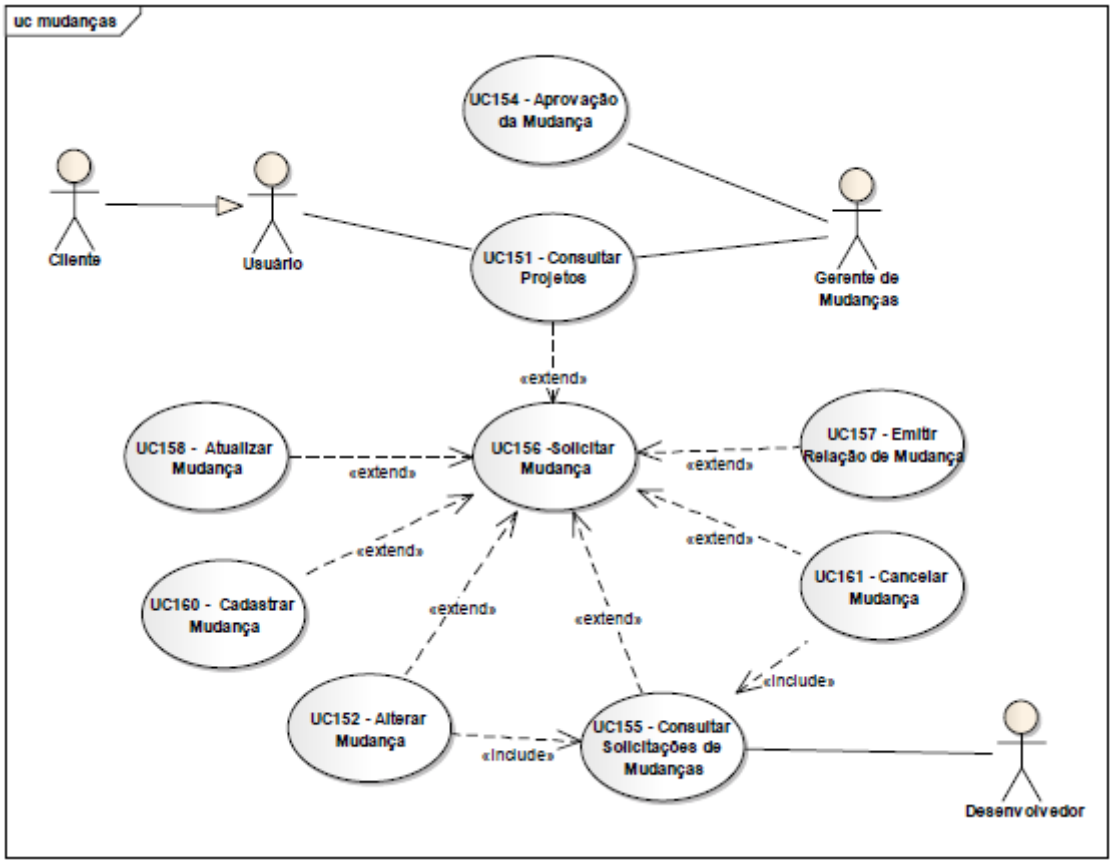

Figura 6. Casos de Uso do módulo de Gestão de Mudanças

Cada ator possui finalidades especificas no diagrama, pois o processo de Gerenciamento de Mudanças não depende somente de um usuário para poder cadastrar ou excluir uma mudança, depende também de outros atores que desempenham papel fundamental para o gerenciamento, como o Gerente de Mudanças e o desenvolvedor. No diagrama, o Cliente é a pessoa que solicita uma mudança ao Usuário do programa, que é o personagem responsável por cadastrar as mudanças tendo acesso a quase todo o programa. O Gerente de mudanças é o personagem que possui acesso ilimitado ao programa podendo além de cadastrar mudanças, fazer o papel principal do sistema, que é decidir se a mudança vai ser aprovada ou reprovada. E finalmente, após todos os processos, o sistema arquiva a mudança, possibilitando ao desenvolvedor acesso ao sistema para verificar as solicitações de mudanças que já foram aprovadas e que já podem ser implementadas ao projeto do cliente.gestão de testes.

e) Gestão de testes

Já o módulo de gestão de testes ou gestão da qualidade software representa o processo de validação que pode ir desde a definição dos requisitos até quando o sistema é colocado em implantação. Segundo Pressman (PRESSMAN, 2004), a qualidade de software é responsável por definir um conjunto de atividades que irão ajudar a garantir que cada produto de trabalho da engenharia de software exiba alta qualidade. Este módulo contempla todas as etapas e fases do processo de testes, por exemplo, teste de unidade, teste de aceitação e teste de integração e é descrito através do caso de uso da Figura 7.

O conceito de teste de software pode ser compreendido através de uma visão intuitiva ou mesmo de uma maneira formal. Existem atualmente várias definições para esse conceito. De uma forma simples, testar um software significa verificar através de uma execução controlada se o seu comportamento corre de acordo com o especificado.

O processo de testes de software representa uma estruturação de etapas, atividades, artefatos, papéis e responsabilidades que buscam a padronização dos trabalhos e ampliar a organização e controle dos projetos de testes (BARTIE, 2007).

Este processo como qualquer outro processo deve ser revisto continuamente, de forma a ampliar sua atuação e possibilitar aos profissionais uma maior visibilidade e organização dos seus trabalhos, o que resulta numa maior agilidade e controle operacional dos projetos de testes (BARTIE, 2007).

A Qualidade de Software é uma área de conhecimento da Engenharia de Software que objetiva garantir a qualidade do software através da definição e normatização de processos de desenvolvimento. A garantia de qualidade de software (SQA) é o exame minucioso de um artefato de software ou estado do projeto com a finalidade de determinar se há algum desvio com relação aos padrões, diretrizes, especificações, procedimento e planos aprovados, e para recomendar melhorias.

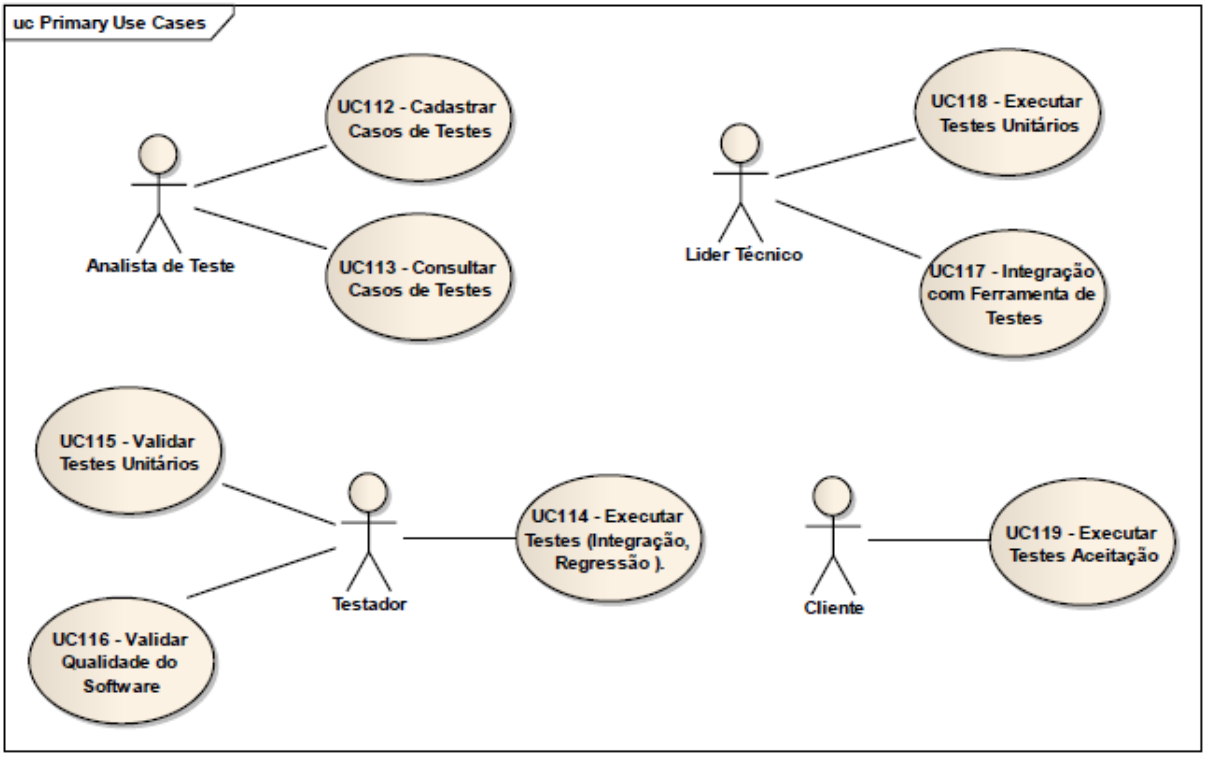

Figura 7. Casos de Uso do módulo de Gestão de Testes

As atividades mais comuns segundo Lewis (2004) no SQA:

- Teste de Software (Software Testing): verifica se requisitos funcionais e não funcionais foram devidamente implementados, entretanto, é que na fase em que ele acontece, muitas vezes é difícil de conseguir alguma qualidade no produto.
- Controle da Qualidade (Quality Control): Monitora se os requisitos estão sendo atendidos satisfatoriamente.
- Gerenciamento de Configuração de Software (SCM - Software Configuration Management): Responsável por identificar, rastrear e controlar mudanças nos elementos do software de um sistema. O SCM controla a evolução do sistema de software, gerenciando versões dos componentes de software e seus relacionamentos.

O módulo de gestão de teste de software tem por objetivo garantir a qualidade do software, com todas as suas funcionalidades, sempre almejando o menor número de erros/bugs. Este módulo é crucial para promover a qualidade do sistema.

f) Gestão de riscos

O módulo de gestão de riscos que compreende na análise dos riscos do projeto. A análise de riscos é crucial para uma boa gestão de projetos e envolve a identificação dos riscos, avaliação dos riscos, disposição dos riscos por ordem de prioridade, estratégias de administração dos riscos, resolução e monitoração dos riscos.

A gerência de riscos é também muito importante, pois com ela pretende-se diminuir os fatores de incidência indesejáveis que podem afetar o desenvolver de um projeto. A necessidade de gerenciar riscos decorre, principalmente, da consciência de existência de fatores, internos ou externos ao projeto, cujo desencadeamento, podem alterar o objetivo do mesmo ao longo do seu ciclo de vida. A identificação desses fatores e/ou das suas causas

constitui uma das etapas fundamentais, de qualquer metodologia de gestão dos riscos (PITMAN, 2000).

Observa-se, diante das pesquisas realizadas, que não existe uma metodologia que possa ser aplicada a qualquer tipo de projeto. Porém, existem varias metodologias voltadas a suportar as atividades de gerenciamento de riscos (MEDEIROS, 2006; PITMAN, 2000).

As funcionalidades desse módulo são descritas através do caso de uso da Figura 8.

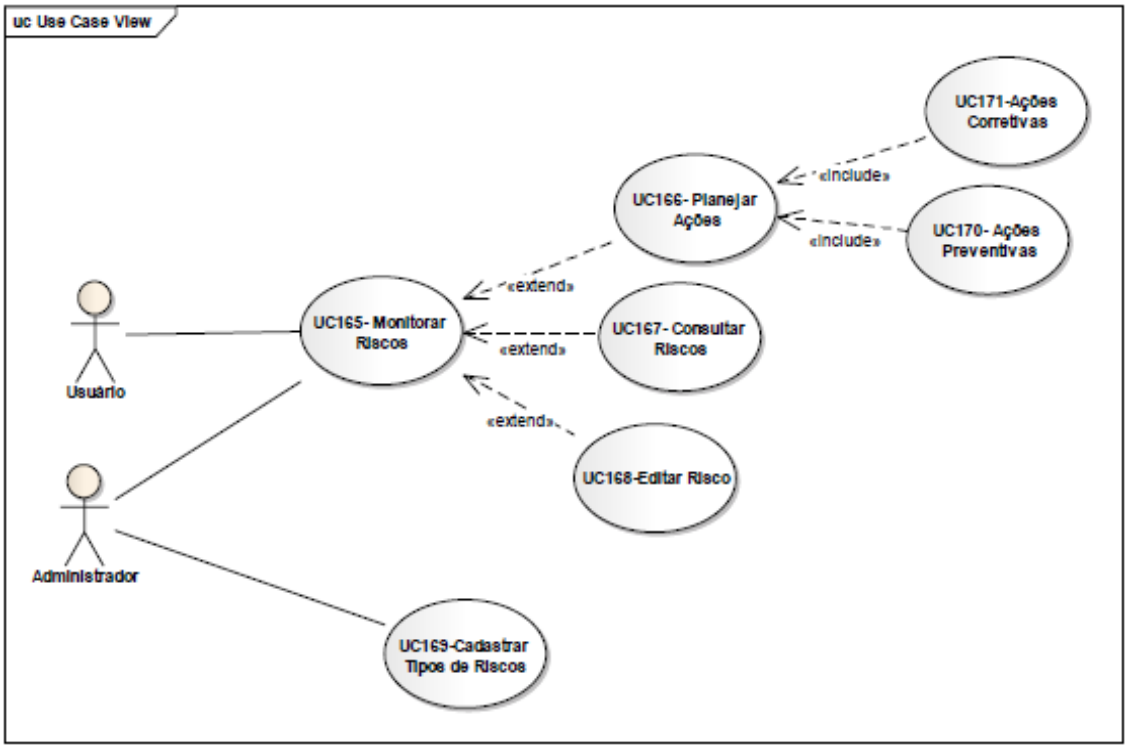

Figura 8. Casos de Uso do módulo de Gestão de Riscos

O Diagrama de Caso de Uso para gerenciamento de riscos tem como atores Usuário e Administrador, que possuem tarefas e restrições diferentes, contendo o Administrador acesso a todo o programa, podendo então administrar todas as suas funcionalidades, como "Monitorar os Risco" e "Cadastrar os tipos de Riscos", e o Usuário possuindo acesso somente a "Monitorar Riscos", pois a função de "Cadastrar Tipos de Risco" está restrita a qualquer tipo de usuário, podendo somente usuários-administradores ter acesso a essa funcionalidade.

g) Gestão de implantação

Uma área relativamente inexplorada no campo da gestão de software é a decisão da implantação ou liberação, decidindo ou não se um produto de software pode ser transferido de sua fase de desenvolvimento para uso operacional. É um *trade-off* entre uma liberação antecipada, para capturar os benefícios de uma introdução no mercado mais cedo, e ou, diferimento do lançamento de produtos, para melhorar a funcionalidade ou qualidade. (PAUL et al., 2004).

Como todos os processos de suporte a serviços, o Gerenciamento de Liberação é suportado pela base de dados para assegurar que infraestrutura seja atualizada. Existem os seguintes tipos de Liberação (Hall, 2003):

- Liberação Completa: Todos os componentes são desenvolvidos, testado, distribuídos e implantados juntos.

- Liberação Delta: composta apenas por itens de configuração que foram modificados desde a última Liberação.
- Liberação empacotada: Liberações independentes e individuais bem como as Liberações completas ou Liberações Delta são combinadas em um só pacote.
- Liberação de Emergência: Liberação de emergência é requerida no caso de dificuldade ou de solução de problema de alta prioridade. Liberação de emergência deve ser utilizada de forma muito reduzida, uma vez que interrompe o ciclo de liberação e é extremamente tendente a falha.

A gestão de liberação é o processo responsável pelo planejamento, programação, armazenamento e liberação de todo o software e hardware autorizado para os ambientes de testes e produção dentro de uma organização. Como os produtos de software estão tipicamente em um ciclo contínuo de desenvolvimento, testes e lançamentos, se torna cada vez mais necessário o aprimoramento dessa gestão. O presente trabalho tem como propósito o desenvolvimento de um módulo de um sistema para gerenciamento de liberação que faz parte de um sistema de gestão de projetos de software. O módulo de gestão de liberação compreende em auxiliar na implantação de um projeto de software e também contempla o gerenciamento dos objetos e artefatos que foram construídos no sistema.

O objetivo principal da Gestão de Implantação é garantir que a integridade do ambiente de produção seja protegida e que os componentes corretos sejam liberados.

As funcionalidades do sistema de gestão de liberação são controlar as liberações do software para os ambientes de testes e produção, armazenar as informações dos artefatos e objetos construídos no projeto de software, bem como o gerenciamento de configuração contendo o armazenamento dos *builds* e versões geradas do software.

Basicamente, o sistema possuirá dois módulos distintos: o módulo de administração que disponibiliza o usuário entrar nas telas de cadastros básicos de implantação, instruções, upload de anexos, além de pesquisas e edições, e o outro módulo de download e aceite por parte do suporte do cliente.

Após o preenchimento de todas as informações e documentações, o suporte responsável por atualizar as novas implantações e/ou versões, poderá liberar o software e concluir a liberação com o aceite.

Cadastro de Implantações: No cadastro de liberações o sistema preencherá automaticamente os campos usuário e data de criação. O desenvolvedor preencherá os campos nome da liberação, código da versão, descrição da versão, nome do projeto referente à liberação, nome da instrução cadastrada também no sistema, status da liberação, que pode ser aberto e concluído, responsável pelo desenvolvimento da liberação, data de abertura da liberação, data de conclusão da liberação, observação e nome do responsável pela implantação no cliente. Além disso, possuirá na tela o botão para upload do arquivo da aplicação que será implantada no cliente e o botão de download para que o suporte do cliente consiga baixar o arquivo e implante a aplicação de maneira confiável e segura.

Pesquisa de Implantações: Na pesquisa de liberações, haverá alguns campos com função de filtros de relatório. Os campos serão nome da liberação, versão, nome do projeto e período de datas de criação, permitindo a verificação das liberações a serem implantadas, podendo estar com status aberto ou concluído.

Edição de Implantações: Para editar uma implantação já cadastrada, o usuário poderá apenas atualizar as informações de status da implantação, observação e responsável pela implantação perante o cliente.

Liberação de Implantações: Após a implantação da aplicação, o suporte do cliente deverá liberar o software e para isso, terá no sistema a tela de liberação, onde o usuário irá informar a liberação, escolher a implantação correspondente além de inserir uma observação de acordo com o andamento da liberação. Haverá também os campos de criado em e criado por, que serão preenchidos pelo próprio sistema.

Cadastro de Instruções: O desenvolvedor poderá cadastrar instruções para que o suporte do cliente utilize na implantação do sistema. Nesse módulo, será necessário preencher o nome da instrução, o plano de contingência, a descrição da instrução. Os campos criado por e criado em serão automaticamente preenchidos pelo sistema.

Pesquisa de Instruções: Na pesquisa de instruções haverá os campos utilizados como filtro de relatório projeto, implantação e nome da instrução. O suporte do cliente poderá pesquisar as instruções que facilitarão na implantação do sistema, ou até mesmo utilizar o plano de contingência caso necessário.

Edição de Instruções: Caso o desenvolvedor ache necessário, haverá possibilidade de editar uma instrução já cadastrada, podendo alterar o plano de contingência e a descrição da mesma.

A Figura 9 mostra os casos de uso do módulo de Gestão de Liberação, juntamente com as associações e seus respectivos atores.

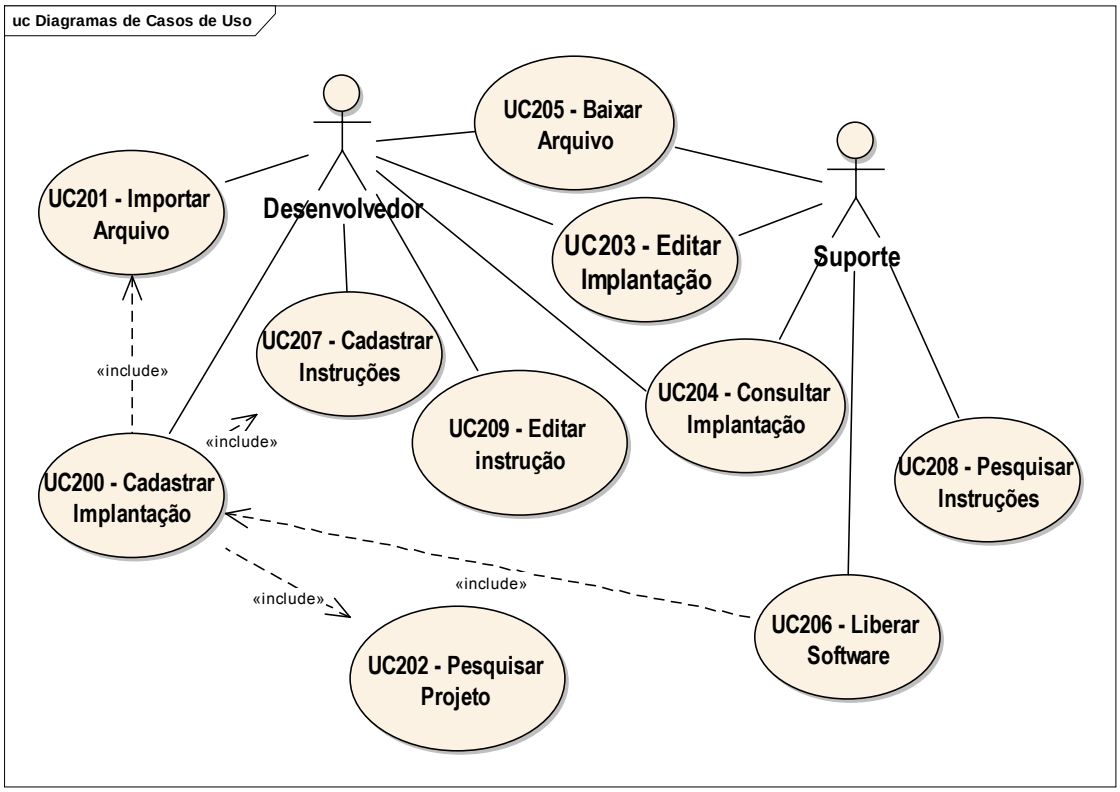

Figura 9. Casos de Uso do módulo de Gestão de Liberação

O sistema de Gestão de Projeto de Software (SGPS) englobará o módulo de Gestão de Liberação. Essa gestão administra e atribui versões de softwares para a produção, com o objetivo de planejar, coordenar e implementar o software, garantindo que haja acompanhamento das liberações, com segurança, integridade e confiabilidade, além de prover documentação, registros, relatórios e instruções.

Outro benefício a ser destacado, é a minimização de possibilidades de implantações incorretas, devido à clareza de informações disponíveis.

3. FLUXO DE APROVAÇÃO E GESTÃO DE CONTROLE DE ACESSO

Para compor o desenvolvimento do fluxo de aprovação do projeto e realizar o gerenciamento de controle de acesso, são utilizados módulos específicos, a saber:

O módulo de *workflow* que será construído para gerenciar todo o fluxo de aprovação da aplicação para todos os estágios de desenvolvimento de um software.

O módulo de *login*, que compreende a autorização e autenticação de usuários no sistema.

O módulo de controle de acesso, que compreende o controle de perfis de usuários e permissões de acesso às ações realizadas no sistema. Por exemplo, o analista de requisitos terá acesso apenas aos módulos: cadastrar requisitos, cadastrar documento de requisitos e inspecionar requisitos e não terá acesso ao módulo de cadastro de projetos, pois somente o gerente de projetos terá essa permissão.

A tela inicial do sistema SGPS é ilustrada na Figura 10.

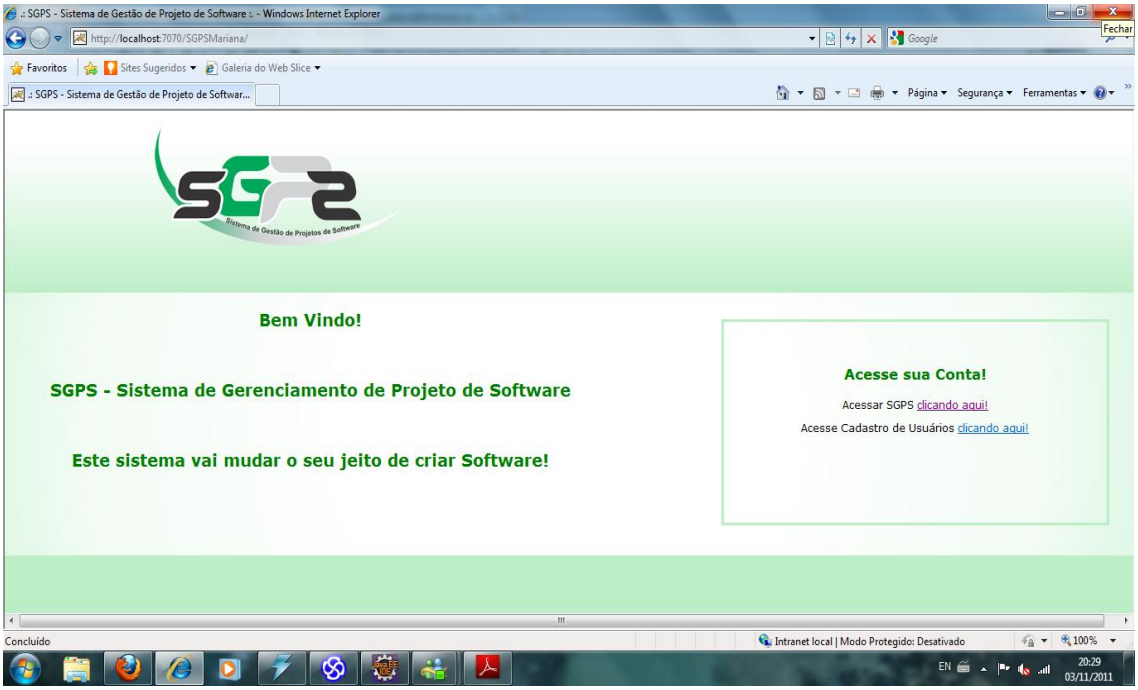

Figura 10. Tela inicial do sistema SGPS

Além disso, outras atividades do gerenciamento de projetos também são contempladas neste sistema como: elaboração de propostas comerciais, seleção e avaliação de pessoal e elaboração de relatórios gerenciais.

4. CONCLUSÃO

Gerenciar bem os projetos dentro das empresas tornou-se não apenas um diferencial competitivo, mas, sobretudo, uma questão de sobrevivência.

Conclui-se que as etapas até então desenvolvidas, contribuirão para consolidar o desenvolvimento do projeto, contemplando informações importantes sobre o mesmo e que serão utilizadas em cada fase de gestão e engenharia de projetos de software.

O sistema SGPS oferecerá para as organizações e/ou instituições de ensino, todo o gerenciamento de projetos de software desde a sua abertura até a sua implantação. Além disso, o sistema de gestão de projetos ajuda a garantir e promover a qualidade do projeto, reduzir seu tempo, maximizar seus recursos e melhorar a comunicação.

O SGPS facilita toda a construção e gestão de projetos de software, aumentando o retorno financeiro, obtido através da redução de custos com projetos em atraso, por exemplo, e a melhoria de processo internos, resultando em aumento de produtividade.

REFERÊNCIAS

BARTIE, A. Processo de Teste de Software – Parte 01. Disponível em: <http://imasters.com.br/artigo/6102/des_de_software/processo_de_teste_de_software_parte_01/>. Acesso em: Janeiro 2011.

BASS, L., CLEMENTS, P., KAZMAN, R. *Software Architecture in Practice*. Second edition, Addison Wesley, 2003.

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. *UML Guia do Usuário*. 2ª Edição (Abrange UML 2.0). Ed. Campus, 2006.

CAMPIONE, M; WALRATH, K. *The Java Tutorial: Object-Oriented Programming for the Internet*. [S.l.]: SunSoft Press., 1996.

DEITEL, Harvey M. and Paul J. DEITEL. *Java: Como Programar*. Sexta edição, Prentice Hall, 2006.

FURLAN, José Davi. Modelagem de objetos através da UML. São Paulo: Makron Books, 1998.

GARCIA, F. P. Como Fazer Estimativas num Projeto de Software. Disponível em: <http://www.dsc.ufcg.edu.br/~patricia/pct/aula9/estimativas.PDF>. Acesso em: Novembro 2010.

GUEDES, G. T. A. *UML 2 - Uma Abordagem Prática*. São Paulo, Editora Novatec, 2009.

LAITENBERGER, O. A Survey of Software Inspection Technologies. Handbook on Software Engineering and Knowledge Engineering, vol. II, World Scientific Pub. Co, 2001. Disponível em: ftp://cs.pitt.edu/chang/handbook/61b.pdf. Acesso em: 13/09/2010.

LARMAN, C. *Utilizando UML e Pa-drões*, 3ª edição, Bookman, 2007.

PAUL, K; RS, Hall. Eureka - Descoberta de Recursos um serviço de componente para a implantação, Anais da Conferência Internacional de Trabalho sobre Implantação de componentes, maio de 2004.

PFLEEGER, *Shari Lawrence.Engenharia de Software Teórica e Prática*. Pearson, São Paulo, 2. ed. 2004.

PITMAN N., PIONE D. *UML 2: Rápido e Prático*. Alta Books, 2006.

PMI - Project Management Institute. *A Guide to the Project Management Body of Knowledge.* – ANSI/PMI 99-01-2004. Project Management Institute. Four Campus Boulevard. Newtown Square. USA, 2004.

PRESSMAN, Roger. *Software Engineering--A Practioner's Approach*. McGraw-Hill, 6th edition, 2004.

PROJECT MANAGEMENT INSTITUTE – Um Guia do Conjunto de Conhecimentos em Gerenciamento de Projetos – Terceira Edição (Guia PMBOK). Four Campus Boulevard: Project Management Institute, Inc, 2004.

SOMMERVILLE, Ian. *Engenharia de Software*. São Paulo, Addison-Wesley, 2006.

STEPHANIE B. *Tutorial do J2EE: Enterprise Edition 1.4*. Rio de Janeiro. Ciência Moderna, 2005.

SOTILLE M. Gerenciamento de Projetos da Engenharia de Software, 2004. Disponível em: http://www.pmtech.com.br/artigos/Gerenciamento_Projetos_Software.pdf. Acesso em: 25/02/2011.

THAYER & DORFMAN. *Software Re-quirements Engineering*. Second Edition, IEEE. Computer Society,1997.

VAROTO, A. C. Visões em Arquitetura de Software. Disponível em: <http://www.ime.usp.br/dcc/posgrad/teses/ane.pdf>. Acesso em: 15/01/2011.