IGOR MUZETTI PEREIRA

Orientador: Tiago Garcia de Senna Carneiro

METODOLOGIA PARA GESTÃO DE PROJETOS DE SOFTWARE EM LABORATÓRIOS DE PESQUISA E INOVAÇÃO EM COMPUTAÇÃO - O CASO DA FÁBRICA DE SOFTWARE TERRALAB

Ouro Preto

Dezembro de 2011

Universidade Federal de Ouro Preto
Instituto de Ciências Exatas
Bacharelado em Ciência da Computação

METODOLOGIA PARA GESTÃO DE PROJETOS DE SOFTWARE EM LABORATÓRIOS DE PESQUISA E INOVAÇÃO EM COMPUTAÇÃO - O CASO DA FÁBRICA DE SOFTWARE TERRALAB

Monografia apresentada ao Curso de Bacharelado em Ciência da Computação da Universidade Federal de Ouro Preto como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

IGOR MUZETTI PEREIRA

Ouro Preto
Dezembro de 2011



UNIVERSIDADE FEDERAL DE OURO PRETO

FOLHA DE APROVAÇÃO

METODOLOGIA PARA GESTÃO DE PROJETOS DE SOFTWARE EM LABORATÓRIOS DE PESQUISA E INOVAÇÃO EM COMPUTAÇÃO - O CASO DA FÁBRICA DE SOFTWARE TerraLAB

IGOR MUZETTI PEREIRA

Monografia defendida e aprovada pela banca examinadora constituída por:

Dr. Tiago Garcia de Senna Carneiro – Orientador
Universidade Federal de Ouro Preto

Dr. Ricardo Augusto Rabelo Oliveira
Universidade Federal de Ouro Preto

Dr. Álvaro Rodrigues Pereira Júnior
Universidade Federal de Ouro Preto

Dr. Joubert de Castro Lima
Universidade Federal de Ouro Preto

Ouro Preto, Dezembro de 2011

Resumo

Este trabalho tem como objetivo a padronização do processo de desenvolvimento de software do TerraLAB - Laboratório para Modelagem e Simulação de Sistemas Terrestres, aliada à implantação de uma metodologia de gestão de projetos. O laboratório TerraLAB, do Departamento de Computação da Universidade Federal de Ouro Preto, é organizado como uma fábrica de software, cujos produtos atendem demandas de pesquisa, desenvolvimento e inovação em geoprocessamento e em modelagem e simulação computacional de processos ambientais. Para isso, os processos de desenvolvimento de software Scrum e Rational Unified Process (RUP) foram combinados à metodologia de gestão de projetos estabelecida pelo Project Management Institute (PMI) e adaptadas à realidade de um laboratório de pesquisa em computação inserido em uma universidade pública brasileira. O contexto acadêmico no qual o laboratório se insere encerra desafios diferenciados daqueles enfrentados por fábricas de software usuais: (i) os colaboradores ainda estão em processo de capacitação e em estágios diferenciados, portanto, não estão prontos para desempenhar qualquer papel que a profissão exija; (ii) os colaboradores não possuem dedicação exclusiva aos projetos, têm como prioridade as atividades acadêmicas; (iii) os colaboradores são em geral pouco comprometidos com os projetos, pois deles independem emprego ou sustento, para alunos de graduação nem mesmo a conclusão do curso está vinculada à projetos bem sucedidos; e (iv) os produtos desenvolvidos são necessariamente vinculados à pesquisas científicas ou à inovações tecnológicas.

Palavras-chave: Rational Unified Process(RUP), Project Management Body of Knowledge(PMBOK), Scrum, Fábrica de Software, Processo de Desenvolvimento de Software.

Abstract

This study aims to standardize the process of software development of TerraLAB - Laboratory for Modeling and Simulation of Terrestrial Systems, coupled with the implementation of a project management methodology. The TerraLAB laboratory, of the Department of Computer Science of Federal University of Ouro Preto is organized as a software factory, whose products serve the demands of research, development and innovation in GIS and computer modeling and simulation of environmental processes. For this, the processes of software development Scrum and Rational Unified Process (RUP) were combined with project management methodology established by the Project Management Institute (PMI) and adapted to the reality of a computer research laboratory embedded in a Brazilian public university . The academic context in which the laboratory is situated, contains different challenges from those faced by usual software factories: (i) the employees are still in process of training and in different stages, therefore, are not ready to play any role that the profession requires; (ii) the employees have no exclusive dedication to the project, their priority are the academic activities, (iii) the employees are generally not very committed to the projects, because they don't need them to employment or support, for graduate students neither the completion of the course is linked to successful projects, and (iv) the products developed are necessarily linked to scientific research or technological innovation.

"E o mais importante, é ter a coragem de seguir seu coração e intuição. Eles de alguma maneira já sabem o que você realmente quer se tornar, todo o resto é secundário". Steve Jobs

Agradecimentos

Agradeço à Deus por me confortar na fé que sinto em seus ensinamentos.

Aos meus pais, irmãos, Glaucy, Tio José Luiz pelas orações e a toda minha família que, com incentivo e apoio, me proporcionaram boas energias para concretizar este trabalho.

Aos Moradores e Ex-Alunos da República Covil dos Gênios pela confiança e por fornecerem ânimo para conclusão dos meus objetivos.

Ao meu professor e orientador Tiago Garcia por seu apoio, paciência e inspiração no amadurecimento dos meus conhecimentos e conceitos que me levaram a execução e conclusão desta monografia.

Sumário

1 Introdução **1**

2 Revisão Bibliográfica **5**

2.1 Fábrica de Software e os passos para sua construção 5

2.2 Metodologia para Gestão de Projetos 8

2.3 Metodologias, Processos e Modelos de Desenvolvimento de Software 12

2.4 Modelos de Melhoria e Normas de Qualidade de Processo de Software 15

2.5 Engenharia de Software Auxiliada por Computador 18

2.5.1 Ferramentas para a Gestão de Projetos 18

2.5.2 Ferramentas de Gerência de Configuração 20

2.5.3 Ferramentas de Documentação de Projetos de Software 24

3 Metodologia **26**

3.1 Nicho de atuação da fábrica de software TerraLAB 27

3.2 Estrutura organizacional e papéis do TerraLAB 28

3.3 O modelo de desenvolvimento de software/modelo do TerraLAB 32

3.4 Ciclo de vida de um projeto TerraLAB 34

3.5 Gestão de projetos do TerraLAB 36

3.6 Processo de desenvolvimento de software/modelo BOPE 39

3.7 Artefatos de Software TerraLAB 42

3.8 Ferramentas CASE utilizadas pelo TerraLAB 43

3.9 Infraestrutura de hardware e software no TerraLAB 44

3.10 Experimentos: Implantação do processo BOPE no TerraLAB 45

4 Resultados **46**

5 Discussão e Trabalhos Futuros **48**

Referências Bibliográficas **50**

Lista de Figuras

1.1 Painel de Controle do NetProject 3

2.1 Diagrama IDEF0 6
2.2 Estrutura hierárquica de um projeto 9
2.3 Gráfico de Gantt 10
2.4 Áreas de conhecimento segundo o PMBOK 11
2.5 Diagrama das baleias - RUP 13
2.6 Metodologia Ágil Scrum 14
2.7 Controle de Versão de Software 22

3.1 Estrutura organizacional do TerraLAB 29
3.2 Diagrama de esforço do modelo de desenvolvimento de software/modelo do TerraLAB 32
3.3 Modelo de Cronograma TerraLAB 37
3.4 Processo de vendas do TerraLAB 40
3.5 Atividade 1: reúne com o cliente 40
3.6 Semelhança entre BOPE e Scrum 41
3.7 Artefatos padronizados do TerraLAB 42

Lista de Tabelas

2.1 Níveis do CMMI. 16
2.2 Níveis do MPS.BR. 17

3.1 Papéis e funções dos membros do TerraLAB. 30

Capítulo 1

Introdução

Este trabalho aborda a gestão de projetos em laboratórios de pesquisa em Computação situados em instituições de ensino e pesquisa. O principal objetivo é desenvolver, implantar e avaliar um processo padronizado para gestão e execução de projetos de Pesquisa, Desenvolvimento & Inovação (PDI) desses laboratórios. O laboratório TerraLAB - Laboratório para Modelagem e Simulação do Sistema Terrestre, do Departamento de Computação, da Universidade Federal de Ouro Preto, é adotado como estudo de caso e visto como uma fábrica de software que diante da necessidade de solucionar suas demandas nesses três temas, decide adotar métodos e técnicas de Gestão de Projetos e de Engenharia de Software para garantir a produtividade de seu time e a qualidade de seus produtos. Atualmente, o laboratório possui três grandes projetos de software em desenvolvimento, alguns em estágio operacional com diversos usuários espalhados pelo mundo, nos quais trabalham aproximadamente trinta desenvolvedores e que somam aproximadamente um milhão de reais investidos em três anos. Entre esses projetos destacam-se os o ambiente de modelagem ambiental TerraME (www.terrame.org) e o sistema de inteligência de negócios aplicada ao planejamento de cidades SIGHabitar (www.terralab.ufop.br).

O contexto acadêmico no qual esses laboratórios se inserem, traz consigo desafios diferenciados daqueles encontrados pelas fábricas de software usualmente encontradas no mercado: (i) os colaboradores que nele trabalham ainda estão em processo de capacitação, são em geral alunos de graduação e pós-graduação; (ii) os colaboradores não possuem dedicação exclusiva, em geral, são bolsistas que trabalham em regime de 12 a 20 horas semanais; (iii) os colaboradores são em geral pouco comprometidos com o sucesso dos projetos, pois dele não dependem seu emprego ou sustento; (iv) os softwares produzidos estão necessariamente atrelados à pesquisa científica e à inovação tecnológica. Por essas razões, as melhores práticas de gerenciamento de projetos, e as metodologias de desenvolvimento de software adotadas no mercado precisam ser adaptadas a esta realidade. De uma maneira geral, busca-se desburocratizar processos tradicionais de desenvolvimento de software, como o processo Rational Unified Process (RUP)

[Ivar Jacobson (1999)], eliminando o excesso de documentação e adotar as características que tornam leves e eficientes alguns processos de desenvolvimento ágil, como o Scrum [Schwaber (2004)], sem ignorar o mínimo de formalismo que garantirá o sucesso do projeto mesmo quando a equipe é pouco capacitada, dedicada e comprometida. Dentro dos prazos e orçamentos proposto nos projetos, é preciso ser ágil e depender mais de processos padronizados do que de pessoas para garantir qualidade dos produtos.

A criação de uma fábrica de software inserida em um contexto acadêmico fornece um ambiente de ensino paralelo ao tradicionalmente oferecido pelas universidades. Os alunos lidam com problemas reais que envolvem clientes e fornecedores, desenvolvem sistemas de grande porte que requerem trabalho em equipe, exercitam suas capacidades de comunicação escrita e oral e são expostos a questões não técnicas que os obrigam a tomar decisões como as que aparecem no cotidiano de sua profissão. Desta maneira, eles têm a oportunidade de aprender com seus erros e aplicar a sua experiência em projetos futuros. O aluno adquire vivência nos diversos papéis envolvidos no ciclo de vida de um software e fica familiarizado com as tecnologias e processos em uso nas empresas desenvolvedoras de software [Nokes (2007)]. Além disso, uma fábrica de software nesse moldes atrairia mais estudantes por meio a exposição de tecnologias de ponta, com vários projetos e oportunidades interessantes, mergulhando-os em suas áreas de interesse e os incentivando a uma colaboração multidisciplinar, diminuindo frequentes queixas que existem com relação à falta de oportunidades de estágios e de iniciação científica.

Para que uma fabrica de projetos de PDI ganhe maturidade, é preciso que seus projetos e processos sejam constantemente monitorados, avaliados, controlados e corrigidos. Projetos de construção civil, de desenvolvimento de software ou de pesquisa podem ser geridos segundo um único padrão que oferece indicadores quantitativos que permitem o monitoramento, em tempo-real, do percentual de resultados esperados que foram concluídos, além do tempo e dos recursos que foram consumidos para produzi-los [PMBOOK (2000)]. O Project Management Institute - (PMI - www.pmi.org) é uma entidade sem fins lucrativos que congrega diversos profissionais atuando em gestão de projetos. Esse instituto estabeleceu um padrão de gestão de projetos reconhecido internacionalmente e que é a pratica de mercado atual. A figura 1.1 apresenta o "painel de controle" do sistema de gestão de projetos NetProject (www.netproject.com.br). Nele é possível observar os indicadores de desempenho quantitativo do projeto de desenvolvimento de software "TerraME", no ano de 2010.

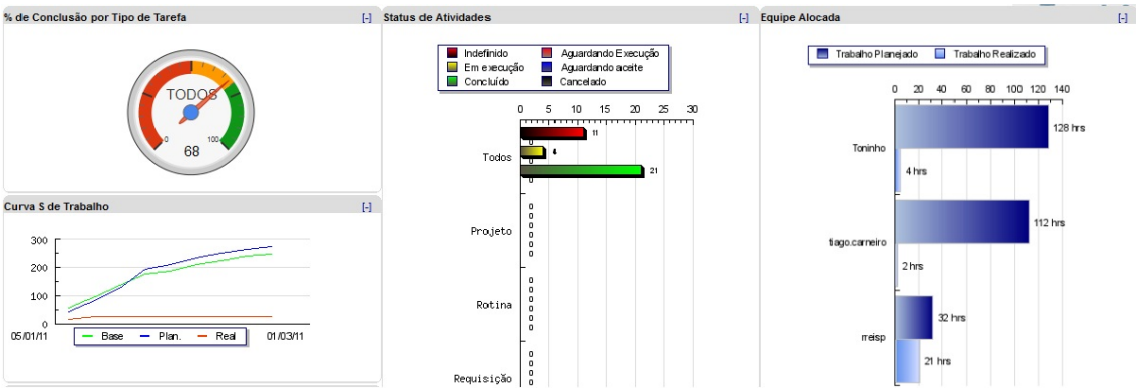


Figura 1.1: **"Painel de Controle" do projeto TerraME no sistema de gestão de projetos NetProject**

No centro do painel de controle, o projeto TerraME tinha 21% de seus resultados concluídos, 4% estavam em execução e 11% estavam aguardando execução. No lado esquerdo superior, é possível ver um resumo do percentual completado do projeto, 68%. No gráfico de linha no lado inferior esquerdo, é possível observar a diferença entre o cronograma planejado e o cronograma executado. No lado direito é possível observar o esforço empreendido por cada recurso do projeto. Infelizmente, quando este gráfico foi gerado a equipe não estava treinada para lançar suas horas de trabalho no sistema de gestão de projetos. Por isso, a "curva S de Trabalho" real não apresentou crescimento ao longo do projeto no gráfico de linha à esquerda. Nem o trabalho realizado pela equipe alocada no projeto foi corretamente monitorado e relatado no gráfico de barra à direita. É importante observar que na fase inicial da implantação e uma metodologia de desenvolvimento e gestão, é demandada uma mudança cultural que implica no aumento da burocracia dos processos de uma fábrica, gerando certa dificuldade de adaptação. Desta maneira, a motivação da equipe é um fator essencial para o sucesso da implantação de novos padrões de processo e métodos. Esta motivação deve permear toda hierarquia de comando da fábrica, desde os analistas de negócios, passando pelos desenvolvedores, até os colaboradores em treinamento. Todos devem estar motivados a implantar os processos para obter uma maior produtividade e garantir a qualidade dos softwares produzidos. Todos devem estar dispostos a implementar métricas que permitam avaliar qualitativa e quantitativamente o desempenho dos projetos.

A concepção de uma fábrica de software está fortemente ligada a aspectos da Engenharia de Software que definem metas a serem alcançadas, como processos de desenvolvimento são bem definidos, a busca pela melhoria contínua desses processos, alocação de perfis funcionais adequados às funções, reuso de código, controle de qualidade, gerenciamento de configuração e boa comunicação. Este trabalho analisa as questões consideradas relevantes durante o desenvolvimento e a implantação da metodologia de gestão de projetos e da padronização do

processo de desenvolvimento do TerraLAB. Ele define de forma clara e objetiva os papéis desempenhados por cada colaborador das equipes de desenvolvimento dos projetos de software. O ciclo de vida dos softwares e dos diferentes artefatos produzidos durante os processos de fabricação são mapeados e documentados, para posterior aplicação em contextos similares. A organização dos projetos do laboratório serviu para validação da metodologia e processos propostos e os resultados obtidos são registrados. As lições aprendidas nesses experimentos serão discutidas. Diversas ferramentas de apoio e automação dos processos de desenvolvimento foram avaliadas e utilizadas na implementação da metodologia desenvolvida. O papel e importância de cada ferramenta são discutidos.

Este texto está organizado da seguinte maneira: o capítulo 2 apresenta uma revisão bibliográfica que cobre os principais conceitos necessários para o entendimento do trabalho. O capítulo 3 informa quais recursos e métodos são empregados para transformar os recursos nos resultados esperados do projeto, atingindo as metas e objetivos específicos, o capítulo 4 apresenta a discussão e os trabalhos futuros, depois das referências bibliográficas são apresentados em anexos os diagramas do processo de desenvolvimento BOPE e os artefatos de software que servem de modelo para serem utilizados na fábrica.

Capítulo 2

Revisão Bibliográfica

Esta seção de texto apresenta os conceitos básicos que fundamentam e permitem o entendimento deste trabalho. Os principais trabalhos correlatos encontrados na literatura são descritos.

2.1 Fábrica de Software e os passos para sua construção

O termo fábrica de software surgiu entre os anos de 1960 e 1970 nos Estados Unidos e Japão, como uma analogia aos processos tradicionais de manufatura, na qual setores distintos são responsáveis por construir componentes específicos de um mesmo produto que é complemente formado até o final de uma linha de montagem. Cusumano (1991) foi um dos primeiros autores a divulgar o termo depois de suas pesquisas relativas às práticas de desenvolvimento de software no final da década de 1980. Segundo o autor, o sucesso das Fábricas de Software no Japão se deve à alta taxa de reutilização de componentes de software, modularização dos sistemas, uso de ferramentas de controle e gerenciamento dos projetos, aumentando assim a produtividade e flexibilidade dos sistemas produzidos.

De acordo com Kent Swanson (1991) a mudança no paradigma de desenvolvimento de software artesanal para científico é identificado pelo uso de ferramentas e métodos padronizados e pelo uso de processos de desenvolvimento de software bem planejados, executados, controlados e avalidados que, passam a permitir o desenvolvimento automatizado de software baseados componentes reutilizáveis. Neste trabalho o termo Fabrica de Software é entendido como uma estrutura organizacional que busca desenvolver produtos de software para um nicho específico a partir do reuso de modelos, padrões e componentes de software bem definidos e testados, uma contraposição direta à maneira artesanal na qual produtos de software podem ser desenvolvidos. Para garantir a alta produtividade, em geral, uma fábrica segue a metodologia de desenvolvimento de software denominada Engenharia Dirigida por Modelo (MDE

- Model-Driven Engineering) ou Desenvolvimento Dirigido por Modelo (MDD - Model Driven Development), na qual modelos e padrões específicos de um domínio de conhecimento são reutilizados por meio de uma linguagem de alto-nível para a construção de aplicações. Desta forma, setores da fábrica ocupam-se do desenvolvimento de arcabouços (frameworks) de software cujos componentes são reutilizados e customizados por meio de uma linguagem de domínio específico (DSL - Domain Specific Language) para o desenvolvimento de aplicações de interesse do usuário final ou cliente [Husu (2006)].

Um equívoco comum é pensar que o paradigma de fábrica de software de alguma maneira inibe ou exclui a criatividade do processo de desenvolvimento de software. Na verdade, o objetivo é tornar mecânica a construção de soluções para problemas recorrentes, automatizando ao máximo o ciclo de vida dos projetos de software, de forma a permitir que o desenvolvedor tenha tempo para criar soluções inovadoras para uma aplicação específica e, portanto, para o usuário final. Ele não deve consumir tempo re-concebendo soluções que são rotineiramente utilizadas pela fábrica.

Luzia Nomura (2007) propõem um modelo para a construção e padronização dos processos de desenvolvimento de uma fábrica de software. Esse modelo baseia-se no processo Rational Unified Process (RUP) e no referencial teórico ofercido pelos diversos autores citados em seu artigo. Este modelo é representado pelo diagrama IDEF0 - Integration DEFinition Language [IDEF0 (2011)] apresentado na figura 2.1.

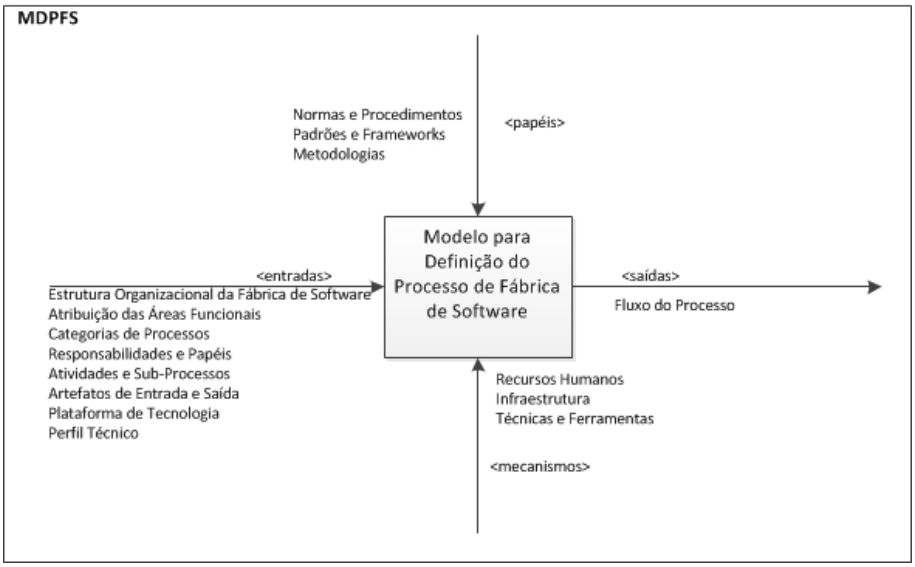


Figura 2.1: **Diagrama IDEF0 representando o processo de construção de fabrica proposto por Nomura at al, 2009.**

O diagrama IDEF0 contém entradas, saídas, papéis e mecanismos envolvidos no processo para criação de uma fábrica de software denotados como setas. As entradas representam elementos fundamentais para a definição do processo da fábrica de software, são constituídas por: (i) Obtenção da visão da estrutura organizacional da fábrica, identificando a postura da organização como sendo funcional ou matricial; (ii) Atribuições das áreas funcionais, definindo os papéis e funcionalidades das áreas envolvidas no processo de produção; (iii) Categorização dos processos, como desenvolvimento de novos produtos, processo de melhoria dos produtos, desenvolvimento de componentes, manutenção corretiva; (iv) Definição dos papéis e responsabilidades individuais que atuam na fábrica ou por grupos de indivíduos; (v) Definição das atividades e sub-processos que todos os papéis desempenham, identificando os artefatos de entrada e saída, determinando a ordem de execução dos sub-processos; (vi) Planejamento do que deve conter exatamente cada artefato; (vii) Planejar a plataforma tecnológica a ser utilizada, identificando o ambiente tecnológico que a fábrica de software deve operar; (vii) Identificação dos perfis técnicos dos profissionais com conhecimento nas áreas de domínio do negócio, plataformas tecnológicas e linguagens de programação.

No diagrama IDEF0, os papéis são entradas que se referem à elaboração de regras para definição da política da organização, podendo as regras ser de natureza administrativa, processuais e técnicas: (i) Construção de normas e procedimentos que contenham as regras; (ii) Utilização de padrões e frameworks como modelos, técnicas e arquiteturas usadas para contribuir na automação do processo; (iii) Utilização de metodologias que se traduzem em modelos de processos bem definidos e implementados como políticas da organização. Os mecanismos também são entradas no diagrama. Porém, se referem às técnicas, ferramentas, recursos humanos e infra-estrutura que constituem o ambiente tecnológico e operacional aos processos de desenvolvimento.

Na figura 2.1, a saída do modelo de definição dos processos de fabricação é representada pelo mapeamento dos fluxos dos processos, contendo a identificação, descrição dos objetivos, representações gráficas e especificações textuais de todas as entradas, papéis e mecanismos definidos.

2.2 Metodologia para Gestão de Projetos

O PMBOK - Project Management Body of Knowledge - é um guia de orientações que integra o conhecimento necessário à gerência de projetos, cujo objetivo é identificar e descrever conceitos e práticas da gerência de projetos em geral, padronizando a terminologia e os processos adotados nesta área de estudo [PMBOOK (2000)]. Esse documento foi produzido e é periodicamente atualizado pelo PMI - Project Management Institute - (www.pmi.org), uma entidade internacional sem fins lucrativos que congrega profissionais atuando na área de gerência de projetos. Por todo o mundo, as principais empresas de desenvolvimento de software e de engenharia gerem seus projetos de acordo com o PMBOK.

Segundo o PMBOK, um projeto é um esforço temporário empreendido para criar produtos, serviços ou resultados exclusivos, doravante denominados entregáveis. Todos os projetos possuem informações importantes para delimitar os resultados esperados do projeto e para gerir as expectativas do cliente com relação a esses resultados: escopo e justificativa. O escopo apresenta o nome dos produtos que serão construídos, descreve a missão dos produtos esclarecendo seus valores para os clientes e usuários. Ele também fornece o escopo-negativo do produto a fim de evitar falsas expectativas por parte do cliente, isto é, descreve suas limitações ou aquilo que o sistema não fará. A justificativa identifica os beneficiados pelo produto e enumera os benefícios obtidos por cada um.

Esses entregáveis demandam que determinadas sejam realizadas para a sua fabricação. Estas atividades por sua vez demandam que recursos sejam para elas alocados. Recursos podem ser bens de consumo, equipamentos, dinheiro ou pessoas. Toda atividade possui um responsável por sua execução. A figura 2.2 ilustra a estrutura geral de um projeto.

#	Nome da Atividade	Duração Planejada (dias)	Status	Responsável Execução	Responsável Aprovação	Predecessor
0	⊟ MeuProjeto	5	Em execução	tiago.carneiro	tiago.carneiro	
1	⊟ Entregavel01 (Componente de Software 01)	5	Em execução	joao.correa	tiago.carneiro	
2	Atvidade01	3	Em execução	joao.correa	tiago.carneiro	
3	Atividade02	2	Aguardando Execução	joao.correa	tiago.carneiro	2
4	Atividade03	1	Aguardando Execução	neli.fonseca	joubert.lima	
5	⊟ Entregavel02 (Componente de Software 02)	4	Em execução	juliana.moreira	joubert.lima	
6	⊟ Sub-entregavel01 (Binario executavel)	3	Em execução	juliana.moreira	joubert.lima	
7	Atividade04	1	Em execução	juliana.moreira	joubert.lima	
8	Atividade05	2	Aguardando Execução	juliana.moreira	joubert.lima	7
9	⊞ Sub-entregavel02 (Manual do Usuário)	4	Em execução	tarik.rocha	tiago.carneiro	

Figura 2.2: **Estrutura hierárquica de um projeto representada na forma uma árvore de itens. No primeiro nível o projeto como um todo. No segundo nível os entregáveis. As atividades (ou processos) aparecem nos itens folha da árvore. Níveis intermediários representam sub-entregáveis.**

Quando as atividades de um projeto são organizadas no tempo e por ordem de dependência, de forma a não super-alocar os recursos dos quais dependem, essas atividades formam o cronograma de execução do projeto, como ilustra a figura 2.3.

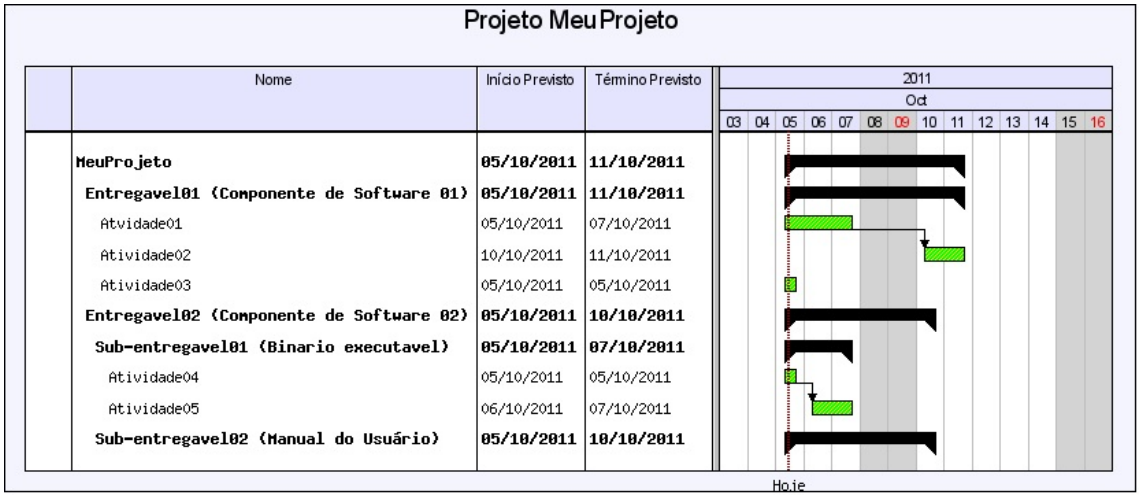


Figura 2.3: **Gráfico de Gantt que ilustra o cronograma de um projeto, apresentando datas de inicio e fim estimadas para cada atividade, além da ordem de precedência das atividades.**

Portanto, todo projeto é um esforço único, que possui datas de início e fim definido, que tem por objetivo produzir entregáveis que sob algum aspecto jamais foram construídos, realizado por uma equipe dedicada e que possui orçamento definido. Desta forma, a gestão de projetos pode ser definida como a aplicação de conhecimentos, habilidades e técnicas para o controle das atividades e dos recursos de um projeto a fim de produzir com qualidade seus entregáveis, no prazo e orçamento previstos. O gerente de projetos é o principal responsável pela realização dos entregáveis de um projeto e, portanto, por sua gestão.

As atividades e recursos do processo de Gerência de Projetos devem ser considerados no planejamento de qualquer projeto. Devem ser claros os resultados esperados com a efetiva implementação do processo de gestão. O processo de gestão é organizado em nove áreas do conhecimento no PMBOK, conforme ilustra a figura 5.

A gerência de projetos pode acontecer de forma qualitativa e quantitativa. Na forma qualitativa, a principal preocupação é a qualidade e impactos dos resultados produzidos pelo projeto. Na forma quantitativa, a principal preocupação é manter o esforço necessário para a realização do projeto dentro do cronograma e custo previsto para o mesmo.

Um projeto é, em geral, realizado em fases que são realizadas um após a outra, sendo necessário que uma fase termine para que outra tenha seu início autorizado pelo gerente. Pro-



Figura 2.4: **Áreas de conhecimento segundo o PMBOK.**

jetos de software são comumente desmembrados nas seguintes estas fases: análise de negócio, concepção, planejamento, construção, testes, implantação e manutenção.

2.3 Metodologias, Processos e Modelos de Desenvolvimento de Software

Um processo de desenvolvimento de software é um conjunto de atividades, parcialmente ordenadas, cuja finalidade é produzir eficientemente componentes de software de boa qualidade. Há tempos a indústria vem buscando padronizar um processo passível de reprodução e que tenha uma boa relação produtividade/qualidade. Essa busca resultou na proposta de vários modelos de processos de software: Modelos em Cascata e Modelo Interativo [Craig Larman (2003)].

Um modelo de processo pode ser visto como uma representação abstrata do esqueleto de processo, incluindo tipicamente algumas atividades chaves, a ordem de precedência entre elas e, opcionalmente, artefatos requeridos e produzidos. Em suma, um modelo de ciclo de vida do projeto que descreve uma filosofia de organização de atividades, estruturando as atividades do processo em fases e definindo como elas se relacionam. O Modelo em Cascata é o paradigma mais antigo da Engenharia de Software, sugere uma abordagem sistemática e sequencial para o desenvolvimento de softwares que começa com a especificação dos requisitos pelo cliente e progride ao longo das fases de planejamento, modelagem, construção e implantação, culminando na manutenção progressiva do software acabado. O Modelo Incremental combina elementos do modelo em Cascata aplicado de maneira iterativa, na qual a cada ciclo de desenvolvimento uma versão melhorada do software é entregue ao cliente.

Uma metodologia de desenvolvimento de software providencia uma estrutura conceitual para reger todas as fases de um projeto de software e pode ser empregada em diversos processos de desenvolvimento. Por exemplo, metodologias de desenvolvimento estruturadas, metodologias de desenvolvimento orientadas por objetos [Sircar et al. (2001)], metodologia dirigida por modelos [Tim Sloane (2003)].

Existem diversos processos de desenvolvimento de software propostos na literatura. O mais tradicional é conhecido como RUP - Rational Unified Process - que consiste em um conjunto de técnica definidas pela empresa Rational Software Corporation para garantir produtividade e qualidade aos projetos de software [RUP (2006)]. Trata-se de um processo proprietário, adquirido posteriormente pela empresa IBM, que se baseia no uso do paradigma orientado por objetos e no uso intensivo da notação UML - Unified Modeling Language - para a construção artefatos formais e padronizados a cada fase de um projeto [Larman (2007)]. Por exemplo, a lista de requisitos e a descrição de casos de uso são artefatos que documentam a concepção de um software, os diagramas de classe, de sequência e de estados são artefatos que documentam o projeto de um software.

O RUP é um processo interativo e incremental, na qual a cada ciclo de desenvolvimento uma versão melhorada do produto é concebida, projetada, construída, testada e entregue ao

cliente. Define como orientações mestras o foco na gestão dos requisitos de software, o uso de arquiteturas baseadas em componentes, o uso de ambientes de desenvolvimento visuais, a verificação da qualidade de software e o controle de mudanças do software. Apesar de poder ser customizado, devido ao excesso de artefatos formais é considerado por muitos um processo pesado e burocrático, adequados a projetos de grande porte.

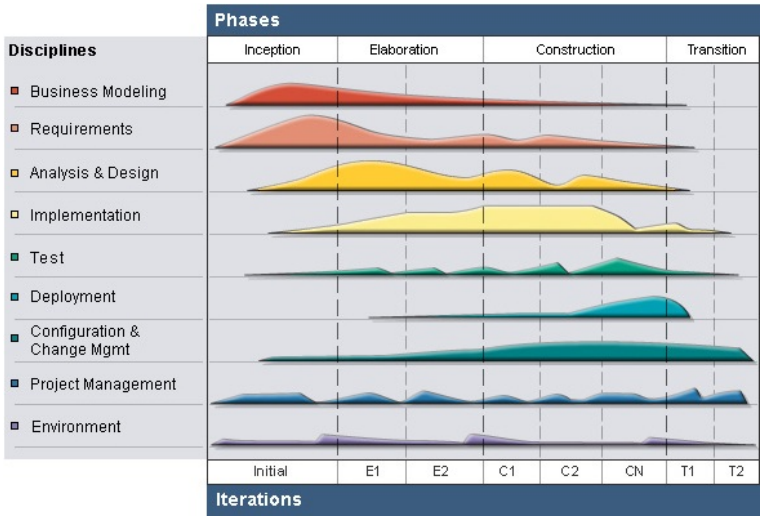


Figura 2.5: **Diagrama das baleias (esforço) do processo Rational Unified Process (RUP).**

Na figura 2.5, pode-se observar que o eixo horizontal representa uma dimensão dinâmica que indica a passagem do tempo, e no eixo vertical uma dimensão estática representando as disciplinas que agrupam as atividades pela sua natureza. Nesta metodologia, o projeto passa por quatro fases básicas: (i)Concepção - entendimento da necessidade e visão do porjeto; (ii)Elaboração - Especificação e abordagem dos pontos de maior risco; (iii)Construção - Desenvolvimento principal do sistema; (iv)Transição - Ajustes, implantação e transferência de propriedade do sistema. Apesar de parecer um modelo em cascata, na verdade cada fase é composta de uma ou mais iterações, o que se assemelha a um modelo em espiral. Estas iterações são geralmente de até duas semanas, ou seja curtas e abordam algumas poucas funções do sistema, reduzindo o impacto das mudanças, pois quanto menor o tempo, menor a probabilidade de haver uma mudança neste período para as funções em questão. Visualiza-se que em determinada fase mais se pode fazer uso de uma determinada disciplina do que de outras. Por exemplo, nas iterações iniciais em projetos de software é comum gastar mais tempo em requisitos e nas iterações posteriores gasta-se mais tempo com a implementação.

Os processos de desenvolvimento ágil são incrementais e iterativos. Porém, adotam ciclos de desenvolvimento curtos para minimizar os riscos dos projetos, enfatizam o uso de comuni-

cações em tempo real e face a face, ao invés do uso de documentos formais. Privilegia times multifuncionais e auto-organizáveis [Edes Garcia da Costa Filho (2005)]. Segundo o "Manifesto Agile", que introduziu o termo em 2001, o desenvolvimento ágil valoriza: (i) Indivíduos e interações mais que processos e ferramentas; (ii) Software em funcionamento mais que documentação abrangente; (iii) Colaboração com o cliente mais que negociação de contratos; e (iv) Responder a mudanças mais que seguir um plano. As metodologias ágeis envolvem menos documentação e mais atividades orientadas a código. É importante observar que o manifesto ágil não rejeita processos e ferramentas, documentação, negociação de contratos, nem planejamento. Porém, argumenta que estas questões têm importância secundária quando comparados com os indivíduos, com o software executável, com a colaboração dos clientes e as respostas rápidas às mudanças [Manifesto for Agile Software Development (2001)].

Entre os processos ágeis, este trabalho destaca a processo Scrum, ilustrando na figura 2.6, por ele aceitar que o desenvolvimento de software é permeado por questões imprevisível e de difícil controle, assumindo que mudanças são a regra ao invés de exceções. Ele se destaca dos demais métodos ágeis pela maior ênfase dada ao gerenciamento de projetos, reunindo atividades de monitoramento e realimentação, em geral, reuniões rápidas e diárias.

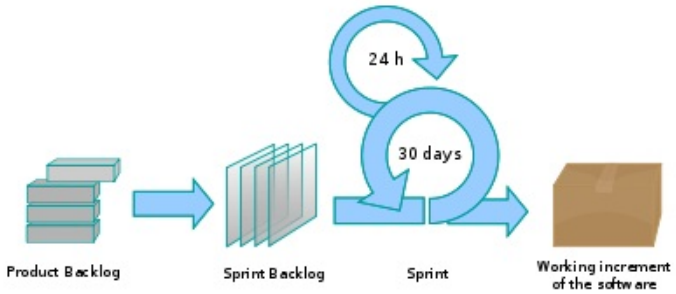


Figura 2.6: **Scrum - Processo de desenvolvimento de software ágil com marcos de projetos mensais e reuniões de acompanhamento diário. O backlog representa o esforço, ou seja, o volume de trabalho que deve ser realizado para produzir os resultados do projeto. O backlog é dividido em sprints, isto é, etapas nas quais algum marco de projeto é atingido. A cada sprint de desenvolvimento versões melhoras dos produtos são entregues aos clientes. Fonte[pt.wikipedia.org/wiki/Scrum].**

2.4 Modelos de Melhoria e Normas de Qualidade de Processo de Software

No passado, a indústria de software percebeu que deveria ser mais produtiva, aumentar a qualidade dos produtos de software, diminuir o esforço e custo dos projetos para se tornarem mais lucrativas. Vários modelos de melhoria de processos de desenvolvimento e várias normas de qualidade foram propostos nesse intuito. As empresas passaram a ser certificadas nesses modelos e normas, estabelecendo um padrão de produtividade e qualidade internacional, permitindo que os níveis de maturidade dos processos dessas empresas pudessem ser definidos e comparados.

As normas da série NBR ISO 9000 foram desenvolvidas para apoiar organizações de todos os tipos e tamanhos na implementação e operação de sistemas eficazes de gestão de qualidade. A norma NBR ISO/IEC 12207 é utilizada pela indústria de software para apoiar uma certificação ISO 9000. Ela foi criada pela ISO (Institute of Organization for Standardization) em conjunto com o IEC (International Electrotechnical Commission) [ISO 9000 (2005)]. A IEC 12207 provê um processo que pode ser utilizado para definir, controlar e melhorar o desenvolvimento de software, ela tem como objetivo central estabelecer uma estrutura comum para o processo de desenvolvimento de software visando a ajudar as organizações a compreenderem todos os componentes presentes na aquisição e fornecimento de software e, assim, conseguirem firmar contratos e executarem projetos de forma eficaz. Seu escopo abrange todo o ciclo de vida do software, desde sua concepção até a descontinuidade do projeto de software, e por todos os envolvidos com produção, manutenção e operação do software, seus processos são agrupados de acordo com sua natureza e resulta em três classes de processos: Processos Fundamentais, Processos de Apoio e Processos Organizacionais [IEC 12207 (1998)].

O modelo de melhoria de processos CMM(Capability Maturity Model) pode ser definido como sendo uma fusão de melhores práticas para avaliação de maturidade de desenvolvimento de software em uma organização. O CMM não diz exatamente como fazer, mas sim o que deve ser feito, por isso conhecido como um modelo e não uma metodologia. Com relação aos principais elementos de um processo de desenvolvimento de software, o CMM descreve estágios de maturidade que passam as organizações enquanto evoluem no ciclo de desenvolvimento de software, através de avaliações contínuas, identificação de problemas e ações corretivas, dentro de uma estratégia de melhoria de processos. Este caminho de melhoria é definido por cinco níveis de maturidade, de 1 a 5, onde o nível 1 é o menos maduro e o nível 5 é o mais maduro [Mark C. Paulk (1993)]:

O MPS.BR(Melhoria de Processo do Software Brasileiro) tem como objetivo definir um modelo de melhoria e avaliação de processo de software, adequado, preferencialmente, às micro, pequenas e médias empresas brasileiras, de forma a atender suas necessidades de negócio e

Nível	**Papel**
5. Otimização	Foco contínuo na melhoria dos processos
4. Quantativamente Gerenciado	Processos são medidos e controlados
3. Definido	Processos são caracterizados para organização e são proativos
2. Gerenciado	Processos são caracterizados por projeto e as ações são frequentemente reativas
1. Inicial	Processos são imprevisíveis, pouco controlados e reativos

Tabela 2.1: Níveis do CMMI.

a ser reconhecido nacional e internacionalmente como um modelo aplicável à indústria de software. Por este motivo, está aderente a modelos e normas internacionais, inclusive cobrindo o conteúdo do CMM. O MPS.BR também define regras para sua implementação e avaliação, dando sustentação e garantia de que é empregado de forma coerente com as suas definições. Ele está dividido em três componentes:

- Modelo de Referência (MR-MPS): contém os requisitos que as organizações deverão atender para estar em conformidade com o MPS.BR. Define, também, os níveis de maturidade e da capacidade de processos e os processos em si;
- Método de Avaliação (MA-MPS): contém o processo de avaliação, os requisitos para os avaliadores e os requisitos para averiguação da conformidade ao modelo MR-MPS;
- Modelo de Negócio (MN-MPS): contém uma descrição das regras para a implementação do MR-MPS pelas empresas de consultoria, de software e de avaliação.

O MR-MPS define sete níveis de maturidade: A (Em Otimização), B (Gerenciado Quantitativamente), C (Definido), D (Largamente Definido), E (Parcialmente Definido), F (Gerenciado) e G (Parcialmente Gerenciado). A escala de maturidade se inicia no nível G e progride até o nível A. Para cada um desses sete níveis de maturidade, foi atribuído um perfil de processos e de capacidade de processos que indicam onde a organização tem que colocar esforço para melhoria, de forma a atender os objetivos de negócio. A tabela abaixo mostra os níveis e processos do MPS.BR [Softex - MPS.BR (2011)]:

Nível	**Processos**
A	Análise de Causas de Problemas e Resolução
B	Gerência de Projetos (evolução)
C	Análise de Decisão e Resolução Desenvolvimento para Reutilização Gerência de Riscos Gerência de Reutilização (evolução)
D	Desenvolvimento de Requisitos Projeto e Construção do Produto Integração do Produto Verificação Validação
E	Avaliação e Melhoria de Processo Organizacional Definição do Processo Organizacional Gerência de Recursos Humanos Gerência de Reutilização Gerência de Projetos (evolução)
F	Medição (para monitoração e controle) Gerência de Configuração Aquisição Garantia da Qualidade Gerência de Portfólio
G	Gerência de Requisitos Gerência de Projetos

Tabela 2.2: Níveis do MPS.BR.

2.5 Engenharia de Software Auxiliada por Computador

Esta seção abordará os sistemas de computação que oferecem suporte aos processos de desenvolvimento de software. Estes sistemas são conjuntamente conhecidos como tecnologias CASE (Computer-Aided Software Engineering, ou Engenharia de Software Auxiliada por Computador) [Pressman (2006)]. Podem ser categorizadas por sua função, isto é, pelos serviços que executam quando utilizadas pelos gerentes e técnicos, ou seja, pelo uso que elas têm nas várias etapas do ciclo de vida de um software. Este trabalho considera a seguinte taxonomia para as ferramentas CASE: ferramentas para gestão de projetos, controle de mudanças, controle de versão, testes e documentação. Uma fábrica de software com processos manuais bem definidos e apoiados por ferramenta CASE é algo que merece ser almejado.

2.5.1 Ferramentas para a Gestão de Projetos

Uma ferramenta de software para gestão de projetos permite o cadastramento de projetos a partir de modelos padronizados por uma organização e oferece serviços de apoio ao planejamento do cronograma de execução e do cronograma de dispensação de recursos. Ela fornece ferramentas pra monitoramento e avaliação, quantitativa e qualitativa, dos resultados e impactos dos projetos. Além de apoiar o fluxo de comunicação e responsabilidade, entre outros serviços e ferramentas. O PMBOK organiza o processo de Gestão de Projetos em nove áreas do conhecimento que estabelecem os requisitos para as ferramentas de suporte à gestão desse processo. Apesar das ferramentas darem suporte às práticas e orientações do PMBOK e serem um ótimo recurso, elas não substituem a capacidade de gerenciar um projeto com sucesso, liderar uma equipe, motivá-la e manter as despesas sob controle [Phillips (2003)].

Para concretizar o efetivo gerenciamento de projetos de software, o usuário da ferramenta poderá tomar suas decisões sobre o projeto com base nos relatórios e gráficos produzidos pelo software através de uma filtragem nas pesquisas, pois não se pode gerenciar o que não se pode medir e analisar. Assim, as ferramentas devem unir em um mesmo ambiente todos os serviços para planejamento e gerenciamento de projetos, bem como, serviços para análises baseadas de métricas de software, sendo a forma mais clara de visualizar os dados produzidos através de gráficos e números [Fabiane Barreto Vavassori (2001)]. Ferramentas como Microsoft Project e NetProject são softwares desenvolvidos para auxiliar o Gerenciamento de Projetos de organizações, elas incentivam e facilitam enormemente a integração e o trabalho colaborativo dos diversos membros das equipes dos projetos. Com o uso de ferramentas deste tipo todos os envolvidos no planejamento e execução dos projetos ficam conectados constantemente, compartilhando informações, registrando atividades realizadas, horas trabalhadas, permitindo uma avaliação precisa da situação real dos projetos.

O NetProject opera tanto em ambientes UNIX quanto Windows, o acesso é totalmente

WEB, sem nenhuma instalação de softwares nas máquinas clientes e com várias possibilidades para customização de contas. O Microsoft Project possui uma interface gráfica simples e amigável, é utilizado desde 1985, é um software popular para gerenciamento de projetos e muitas pessoas que não possuem experiência com ferramentas desta área acabam começando por ele.

2.5.2 Ferramentas de Gerência de Configuração

A Gerência de Configuração de Software (GCS) visa apoiar a evolução de sistemas de software, ela é reconhecida como um elemento crítico do processo de manutenção de software [IEEE (1987)]. O propósito da GCS é estabelecer e manter a integridade de todos os produtos de trabalho de um processo ou projeto e disponibilizá-los a todos os envolvidos, ou seja, para que não haja inconsistências nos artefatos importantes para o projeto, a criação e as modificações nestes artefatos devem ser controladas por outros membros da equipe como os gerentes de configuração por exemplo. Este processo faz parte do ciclo de vida do software e permite que este monitoramento seja executado. Destaca-se dois tipos de ferramentas da gerência de configuração:

2.5.2.1 Ferramentas de Controle de Mudanças

Define-se primeiramente as ferramentas para controle de mudanças no projeto de software, para um grande esforço de desenvolvimento de software, mudanças descontroladas levam rapidamente ao caos. O controle de mudanças combina procedimentos humanos e ferramentas automatizadas para proporcionar um mecanismo de controle de mudanças. Pressman (2006) descreve o processo de controle de mudanças da seguinte maneira:

"Um pedido de mudança é submetido e avaliado quanto ao mérito técnico, potenciais efeitos colaterais, impacto global sobre outros objetos de configuração e funções do sistema, e o custo projetado da mudança. Os resultados da avaliação são apresentados como um relatório de mudança que é usado por uma autoridade controladora de mudanças (Change Control Authority - CCA) - uma pessoa ou grupo que toma uma decisão final sobre o status e a prioridade da mudança. Uma ordem de mudança de engenharia (Engineering Change Order - ECO) é gerada para cada mudança aprovada. A ECO descreve a mudança a ser feita, as restrições que devem ser respeitadas e os critérios de revisão e auditoria. O objeto a ser mudado passa por um check-out (registro de saída) no banco de dados de projetos, a mudança é feita e as atividades de SQA (garantia da qualidade de software) apropriadas são aplicadas. O objeto passa então por um check-in (registro de entrada) no banco de dados e são usados os mecanismos apropriados de controle de versão para criar a nova versão do software".

O objetivo de uma ferramenta de controle de mudanças é ajudar o desenvolvedor a rastrear as mudanças através de uma matriz de rastreabilidade que auxilia na identificação dos impactos ocorridos durante o ciclo de vida do software, entender o porquê de cada uma e qual o seu impacto no projeto como um todo, através dela é possível ter um acompanhamento da evolução do projeto, pois no gerenciamento de projetos de software as alterações e a avaliação do impacto da mudança durante o ciclo de desenvolvimento é essencial e crítico.

O Trac é um sistema de acompanhamento de projetos de desenvolvimento de software que utiliza uma abordagem de acesso via web como algumas ferramentas de gerenciamento de projetos. Ele ajuda a equipe do projeto a desenvolver grandes softwares sem deixá-los sair fora do escopo do projeto, ela fornece uma interface de integração para o Subversion(Ferramenta de Controle de Versão). Funcionalidades importantes nesta ferramenta são a timeline que mostra todos os eventos atuais e passados que foram executados no sistema em ordem, proporcionando uma visão geral do projeto de maneira fácil e o roadmap que mostra as tarefas a serem seguidas separadas por Milestones(Marcos de desenvolvimento) do projeto, além ainda da customização de pesquisas para visualização de tickets, indicando suas resoluções, milestone, proprietário, tipo e vários outros. O Trac é um software livre sob os termos da licença BSD(Berkeley Software Distribution) modificada que pode ser visualizada no site do próprio projeto[http://trac.edgewall.org/].

O Mantis é um outro software livre porém sob as normas da licença GPL(General Public License), essas duas licenças BSD e GPL são muito comuns para o software livre, mas a licença BSD requer apenas o reconhecimento dos autores e algumas pequenas restrições, já a GPL requer que trabalhos derivados sejam licenciados sob a mesma licença, ou seja, sob a GPL. O Mantis é um rastreador de ocorrências e intervenções técnicas capaz de manter e administrar os registros e solicitações dos colaboradores, facilitando o rastreamento de falhas e problemas em potencial, gerando estatísticas e documentando todo o processo de resolução das ocorrências ou problemas e ele apresenta uma interface web para o acesso, mais informações sobre as funcionalidades do sistema no site do próprio projeto [http://mantisbt.org/].

2.5.2.2 Ferramenta de Controle de Versão

O Controle de Versão de Software combina procedimentos e ferramentas para gerenciar diferentes versões de objetos de configuração que são criadas durante o processo de engenharia de software [Pressman 2006]. Um grande número de diferentes abordagens automatizadas para controle de versão foi proposto no decorrer da última década. A diferença fundamental nas abordagens é a sofisticação dos atributos que são usados para construir versões e variantes específicas de um sistema e os mecanismos do processo de construção. Esses sistemas também suportam o conceito de fixação como linha de base (baselining), impossibilitando assim modificações descontroladas de uma versão particular. Faz-se então importante entender a lógica de funcionamento básica desses sistemas.

As ferramentas de controle de versão utilizam o conceito de repositórios, que é um lugar central onde ficam armazenadas as cópias principais de todas as versões dos arquivos do projeto em uso, usa-se então uma máquina servidora para centralizar estes arquivos a fim de possibilitar uma política de backup destes arquivos e disponibilizar maior segurança para os dados. Com isto, o desenvolvedor não trabalha diretamente com os dados originais do servidor,

eles são copiados para sua estação de trabalho através do comando check-out realizado pelo desenvolvedor que, depois de trabalhar com suas cópias, pode enviar novamente ao servidor fazendo um check-in dos arquivos que, gerarão novas versões de determinados documentos.

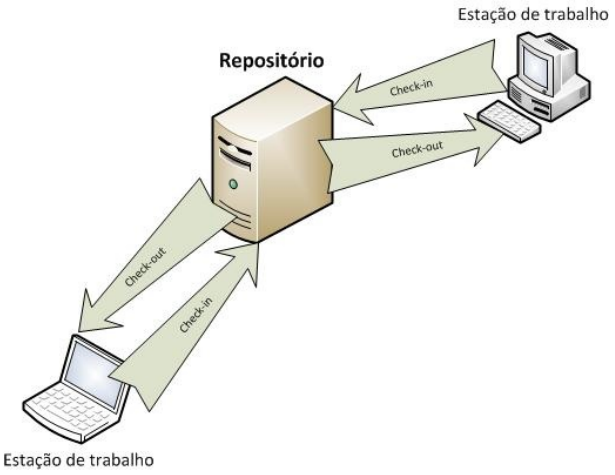


Figura 2.7: **Funcionamento de um Sistema de Controle de Versão**

Apesar de existirem diversas ferramentas de controle de versão, destaca-se neste trabalho dois tipos muito utilizados atualmente que são o CVS(Concurrent Version System) e o SVN(Subversion), ambos são úteis para se controlar versões de um software durante seu desenvolvimento, são de código aberto e normalmente o servidor roda em sistemas UNIX, enquanto seus clientes podem rodar em qualquer sistema operacional através de outras ferramentas como por exemplo o TortoiseCVS[TCVS 2004] e o TortoiseSVN [http://tortoisesvn.tigris.org/].

O TortoiseCVS e TortoiseSVN são ferramentas clientes de subversão integrados ao Windows, ou seja, elas apresentam um conjunto de ações sistemáticas para que o usuário que pode ser um desenvolvedor por exemplo, configure seus arquivos e pastas, visualizando informações de estado, contando com uma sobreposição dos ícones, possibilita acesso local e remoto ao repositório e o segundo possui ainda uma funcionalidade de visualização de diferenças entre dois arquivos por exemplo, chamada DIFF.

Sobre o armazenamento do CVS é feito em arquivos RCS que são criados em toda árvore de diretórios do projeto para versionar arquivo por arquivo, possibilitando maior poder para recuperar possíveis falhas apenas editando esses arquivos, por se tratar de arquivos de disco existem problemas de acesso concorrente e sua velocidade é um pouco comprometida por se usar um sistema de arquivos que é uma maneira do sistema operacional estruturar, organizar e gerenciar os arquivos desde o acesso pelos usuários ao seu conteúdo. Ele gerencia versões diferentes para cada arquivo do projeto, permitindo a criação de ramos(branchs) e etiquetas(tags) por arquivo, ele não permite restaurar a versão do projeto a partir de uma etiqueta específica, o CVS não armazena metadados somente arquivos [http://cvs.nongnu.org/].

O armazenamento do SVN é feito em banco de dados Berkeley DB e existem aplicativos

para recuperação de falhas, o acesso concorrente é transacionado pelo banco de dados, em relação à velocidade ele é mais rápido porém com um problema ao conectar com o repositório e fazer o primeiro checkout, pois ele traz uma cópia local de todos os arquivos. Para cada ramo ou etiqueta cria-se uma cópia no repositório e todos os arquivos do projeto ganham um número de quatro dígitos, o SVN permite restaurar uma versão do projeto a partir de uma etiqueta específica, permite que se vincule e versione também atributos relacionados aos arquivos e se comunica com o sistema de controle de mudanças Trac via um módulo do apache utilizando o protocolo HTTP [http://subversion.tigris.org/].

2.5.3 Ferramentas de Documentação de Projetos de Software

Projetos de software precisam do registro e armazenamento de vários documentos produzidos em tempos distintos e de forma colaborativa. Uma documentação bem gerida é importante para garantir melhores resultados em ambientes cooperativos, pois funciona como uma memória descritiva da pesquisa e do desenvolvimento. Devido à variação do horário de trabalho dos recursos humanos e a descentralização das atividades de desenvolvimento para além da sede do ambiente de desenvolvimento, essas ferramentas devem permitir a construção dos documentos de forma cooperativa via Internet. Essas ferramentas também devem oferecer serviços para pesquisa e recuperação de documentos, de forma rápida e eficaz. É importante que as ferramentas sejam flexíveis a ponto de permitir que os documentos possam ser organizados de maneira estruturada e de acordo com a natureza de cada projeto, além de estabelecer a obrigatoriedade de alguns documentos padrão. Dessa forma, percebe-se que a gestão de documentos torna-se indispensável em um ambiente de desenvolvimento de software.

As Wikis foram criadas e desenvolvidas para que a partir de um navegador web os usuários possam editar suas páginas de um modo simples através de uma linguagem de marcação específica. A vantagem da abordagem de uma ferramenta de documentação colaborativa é que qualquer membro da equipe pode editar os documentos para corrigir erros, adicionar conteúdo e manter sempre tudo atualizado. Elas também são excelentes para descrever passo a passo a instalação de softwares usados pela equipe e de como utilizar determinadas ferramentas, ou seja, armazenamento de tutoriais. Existe também a edição privada, onde o autor edita seus documentos e não deseja que certas pessoas os vejam pelo menos por determinado período de tempo ou para anônimos que acessam a wiki, este caso de uso é obtido através das permissões que este tipo de sistema oferece. Existem muitos softwares wiki disponíveis, os mais conhecidos são o DokuWiki e o MediaWiki, este último usado pelo Wikipédia.

A diferença entre um site wiki e um site "comum", é que os sites wiki oferecem algumas peculiaridades:

- Ferramenta de edição online e coletiva de documentos (textos, mapas, etc) na internet, com uma linguagem simplificada e leve;
- Ferramenta essencial de revisão destes textos, que arquiva as versões das modificações e disponibiliza formas de comparação das diferenças entre as versões;
- Todos os textos editados podem ficar disponíveis online, de imediato, logo após a edição, e revisados a qualquer momento, com as ferramentas de revisão.

De acordo com o próprio site do Dokuwiki sua definição é: *"Um Wiki padronizado e simples de usar, visando principalmente a criação de documentação de qualquer natureza. É destinado às equipes de desenvolvedores, grupos de trabalho e pequenas empresas. Ele possui*

uma sintaxe simples e poderosa que garante que os arquivos de dados sejam legíveis fora do Wiki e facilita a criação de textos estruturados. Todo dado é armazenado em arquivos de texto simples(.txt) - nenhum banco de dados é necessário" [http://dokuwiki.org/pt-br:dokuwiki/].

Capítulo 3

Metodologia

Esta seção apresenta os métodos, técnicas e ferramentas de software utilizadas para a construção da fábrica de software TerraLAB. Além dos processos e artefatos de software definidos e implantados em seu ambiente de produção, os experimentos realizados para aferir melhorias quantitativas e qualitativas em seus processos produtivos são discutidos.

O modelo de definição do processo de fábrica de software apresentado no diagrama IDEF0 da figura 2.1 foi utilizado para a construção do processo de desenvolvimento da fábrica de software TerraLAB. O processo de desenvolvimento RUP foi mesclado com o processo Scrum para dar origem ao processo BOPE (Boosted Object-oriented software Process Engineering). Esse novo processo busca tirar vantagens do formalismo processuais e artefatos padronizados do RUP para garantir qualidade aos artefatos e entregáveis produzidos por uma equipe ainda em capacitação, inexperiente e descomprometida. Além de tirar vantagens das interações curtas, das comunicações face-a-face, da cooperação com o cliente e da capacidade de reação às mudanças do processo Scrum, para manter o processo mais leve, dinâmico e produtivo. A ênfase na gestão de projeto associada às recomendações PMBOK são utilizadas para permitir a gestão quantitativa dos entregáveis produzidos, além de manter os projetos dentro dos prazos e consumo de recursos estimados.

Finalmente o processo BOPE foi implantado em projetos piloto e seus impactos mensurados por meio de experimentos descritos à frente. Durante a implantação do processo, diversas ferramentas CASE foram avaliadas, implantadas e configuradas para automatizar a gestão dos projetos, a gestão de documentação, o controle de versões e a gestão de mudanças.

3.1 Nicho de atuação da fábrica de software TerraLAB

O TerraLAB é uma fábrica de software que tem como nicho de atuação os seguintes temas: "Geoprocessamento" e "Modelagem e Simulação do Sistema Terrestre". Adota-se uma metodologia de desenvolvimento dirigida por modelos (MDD), no qual extensões da linguagem de programação Lua (www.lua.org) são utilizadas como uma linguagem de domínio específico (DSL). Essas extensões são especialmente projetadas para customizar aplicações implementadas sobre os diversos frameworks desenvolvidos e mantidos pelo laboratório: TerraME, TerraVR e TerraGEO. O TerraME é um framework para modelagem e simulação da dinâmica espacial das interações sociedade-natureza. O TerraVR é um framework que estende as funcionalidades do TerraME para permitir o desenvolvimento de modelos dinâmicos georeferenciados em três dimensões, com os quais o usuário interage, em tempo real, por meio de equipamentos de realidade virtual. Entre esses equipamentos destacam-se os visores presos à cabeça, luvas, mouses 3D e joysticks. O TerraSIG é um framework para o desenvolvimento de Sistemas de Informação Geográfica (SIG) baseados na biblioteca C++ TerraLib (www.terralib.org) e na arquitetura de plugins do aplicativo TerraView (www.dpi.inpe.br/terraview/).

3.2 Estrutura organizacional e papéis do TerraLAB

O TerraLAB gere seus projetos de desenvolvimento de software conforme a metodologia proposta pelo PMBOK. A administração por projetos é promovida por uma estrutura organizacional na qual a autoridade é distribuída por projetos, as responsabilidades são delegadas a diferentes níveis hierárquicos dentro da equipe de projeto e o sistema de comunicação é delineado para que as equipes realizem as atividades e exerçam a autoridade que lhes competem, de forma a alcançar os objetivos organizacionais.

No TerraLAB, as responsabilidades, atividades e autoridades são delegadas aos membros das equipes segundos os papéis definidos na tabela 3.1. É importante frisar que uma pessoa pode exercer diversos papéis em um projeto e participar de vários projetos.

A estrutura de gerenciamento de projetos e competências do TerraLAB é de matriz forte, isto é, o gerente de projeto possui maior influência na rotina de trabalho dos colaboradores da fábrica do que os gerentes funcionais das equipes [Leandro Alves Patah (2002)]. Para isso, o gerente de projetos também possui um alto apoio dos níveis organizacionais acima dele e dedicação integral aos projetos. O gerente de projeto é visto como um cliente exigente pelo gerente de fábrica, ele exige prazos mínimos, baixo custo e produtos de qualidade. O gerente de fábrica é o principal responsável técnico pela execução dos projetos. Ele garante a qualidade dos produtos, conhece os perfis dos membros de equipe, delega funções, negocia prazos e recursos para a execução dos projetos e compartilha da forma eficaz esses recursos entre os diversos projetos em andamento. De certa forma, os recursos humanos pertencem à fábrica e não aos projetos.

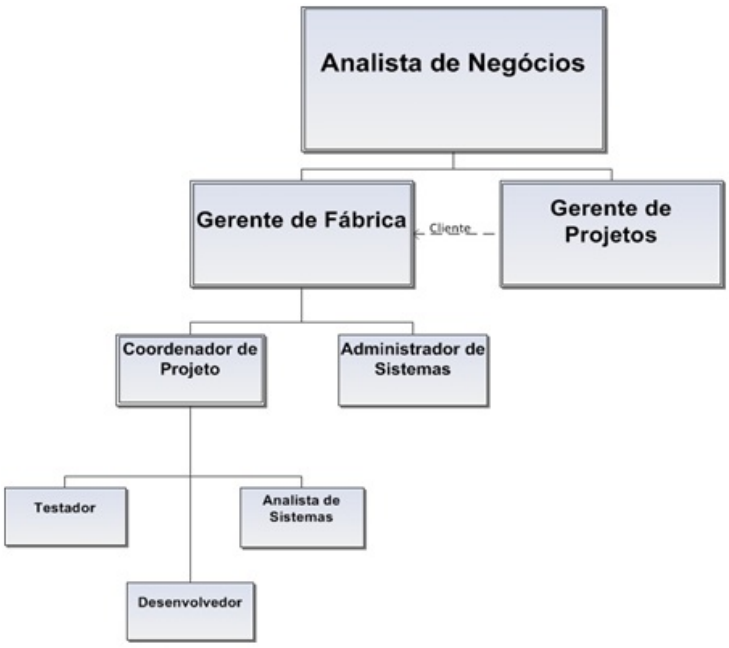


Figura 3.1: **Estrutura organizacional do TerraLAB**

Papel	Direitos e Deveres	SIGHabitar	TerraME
Analista de Negócios (Professor)	Cuida da prospecção do mercado e venda dos serviços. Responsável pelo relacionamento com o cliente, ele deve administrar suas expectativas e interesses. Define escopo, escopo negativo e entregáveis do projeto junto com o cliente. Elabora a proposta técnica/comercial do projeto, na qual define custo, prazo e esforço com o apoio do Gerente de Fábrica e de Projetos.	Tiago	Tiago
Gerente de Fábrica	Responsável por definir e gerenciar o controle de versão e mudanças dos projetos em andamento. Avalia o planejamento do projeto. Gere a qualidade dos entregáveis produzidos, de acordo com revisões realizadas em pontos específicos do ciclo de vida de desenvolvimento. Realiza auditorias de qualidade e coleta métricas ao longo do projeto.	Rodrigo	Rodrigo
Gerente de Projetos	Responsável pelo planejamento e acompanhamento dos cronogramas de atividades e de recursos do projeto. Aloca recursos, mantém a equipe do projeto focada, dimensiona tarefas junto com os coordenadores de projetos. Gere riscos das atividades em desenvolvimento.	Igor	Igor
Administrador de Sistemas	Responsável por manter o ambiente de software de apoio ao processo de desenvolvimento da fábrica. Administra e mantém a infra-estrutura de hardware e software da fábrica. Gerencia usuários, privilégios, *backup*s, etc.	Igor	Igor
Coordenador de Projeto (Arquiteto de Software)	Líder de equipe. Responsável técnico pelo planejamento do projeto. Planeja testes de software junto com os testadores. Orienta os desenvolvedores durante a fase de implementação. Responsável pela gestão do cronograma de atividades junto com o Gerente de Projetos. Responsável pela gestão de mudanças de software junto com o Gerente de Fábrica.	João Tácio (mestrando)	Tiago
Desenvolvedor (Engenheiro de Software)	Responsável pela implementação do software	Francisco, Augusto	Henrique, Antonio
Analista de Sistemas	Responsável pelo levantamento e análise dos requisitos de software, destacando as funcionalidades e delimitações do sistema. Auxilia o testador no planejamento dos testes.	Érika	Pedro, Tiago
Analista de Testes	Planeja e executa os casos de testes para verificar/validar a realização dos casos de uso em relação aos requisitos.	Érika	Pedro

Tabela 3.1: Papéis e funções dos membros do TerraLAB.

Enquanto o gerente de projetos dita ritmo das equipes para que os produtos sejam entregues no prazo e com os recursos disponíveis, o gerente de fábrica premia ou corrige as equipes de projeto para garantir a qualidade desses produtos. Fazendo uma jocosa comparação entre a fábrica de software TerraLAB e as antigas embarcações náuticas em que a tripulação remava na ausência de vento, a equipe de projeto é essa tripulação e é quem efetivamente faz a fábrica se mover, o gerente de projetos dita o ritmo das remadas com seu tambor e o gerente de fábrica usa o chicote.

Tanto o gerente de projeto quanto o gerente de fábrica são subordinados ao analista de negócio que, em geral, é um papel exercido pelo professor que lidera o laboratório de pesquisa onde a fábrica de software se insere. Ele é o principal responsável pela captação de novos negócios. Ele também deve modelar e gerir os negócios do laboratório, de forma a manter sob controle as expectativas dos clientes com relação aos entregáveis que serão produzidos, prazos e custos.

Os coordenadores de projeto lideram as equipes de desenvolvimento. Geralmente, esse papel é desempenhado por alunos de pós-graduação ou bolsistas experientes e já graduados. Eles são os principais responsáveis pelo projeto técnico das soluções, definem a arquitetura e ajudam a conceber as soluções, orientam as atividades de construção, de teste e de implantação. Além disso, no contexto do projeto que lideram, eles apóiam as atividades dos gerentes de projeto e do gerente de fábrica. A figura 3.1 apresenta o organograma que representa a estrutura organizacional da fábrica de software TerraLAB.

3.3 O modelo de desenvolvimento de software/modelo do TerraLAB

O TerraLAB adota um modelo de desenvolvimento de software em espiral e incremental, no qual os projetos evoluem em ciclos composto por uma sequência padronizada de fases. Esse modelo evolucionário combina a natureza iterativa da prototipagem com os aspectos controlados e sistemáticos do modelo em cascata. Ele oferece o potencial necessário para o desenvolvimento rápido de versões de software cada vez mais completas [Pressman, 2006]. Artefatos e versões de produtos são entregues aos clientes durante todo o processo de desenvolvimento, conferindo transparência ao processo e mantendo o cliente informado.

A figura 3.2 apresenta o diagrama de esforço do modelo de desenvolvimento de software/modelos do TerraLAB. Os projetos passam pelas fases dispostas horizontalmente na figura. A cada fase, as atividades de desenvolvimento dispostas verticalmente precisam ser realizadas para que versões incrementais dos resultados esperados sejam entregues ao cliente. Desta maneira, durante a fase de análise de negócio, mais esforço (custo/tempo) precisa ser despendido para concluir os artefatos da atividade de levantamento e análise de requisitos. Enquanto que, uma quantidade menor de esforço será dedicada às atividades de projeto, implementação e teste. Pois, quando necessário, somente protótipos dos entregáveis serão produzido nesta fase.

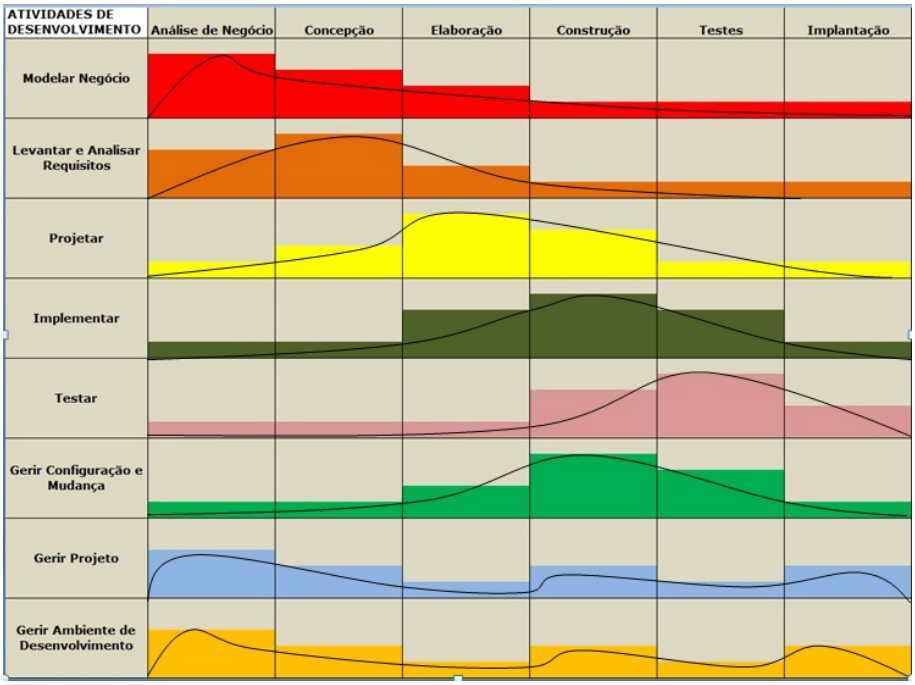


Figura 3.2: **Diagrama de esforço do modelo de desenvolvimento de software/modelo do TerraLAB**

As fases e atividades de desenvolvimento são dimensões ortogonais do modelo de desenvolvimento. Enquanto os projetos atravessam fases que mostram seu estágio de desenvolvimento, as atividades de desenvolvimento determinam quais papéis funcionais precisam existir na fábrica. Esses papéis podem ser desempenhados por especialistas ou equipes de especialistas. Essas equipes são chamadas células de desenvolvimento e possuem um gerente funcional. Alguns dos papéis funcionais mais comuns são: Analista de sistema, programador e analista de teste. Estes papéis podem dar origem à: Célula de gestão de requisitos, célula de desenvolvimento e célula de teste.

3.4 Ciclo de vida de um projeto TerraLAB

Antes de um projeto ter seu inicio autorizado ele precisa ter seu financiamento acordado com um cliente. Para isso, o analista de negócio junto com o futuro coordenador de projeto devem elaborar a proposta técnica/comercial desse projeto, doravante chamada simplesmente de proposta de projeto. Esse documento oferece uma visão de alto nível do projeto que permite manter sob controle as expectativas do cliente e a estimativa de esforço pelos gerentes de fábrica e de projeto. A proposta de projeto define o escopo positivo e negativo do projeto, seu objetivo, os principais entregáveis, os tipos dos entregáveis (software, banco de dados ou modelo de simulação), além dos prazos e custos estimados para cada entregável. Depois de elaborada, a proposta do projeto é submetida para a avaliação do cliente que pode solicitar seu refinamento, recusá-la ou aprová-la. Se aprovada, o início do projeto é autorizado e seu ciclo de vida tem início. No TerraLAB, o ciclo de vida de um projeto é composto pelas seguintes fases:

- **Análise de Negócio:** O objetivo da análise de negócio é limitar o escopo do projeto e realizar estimativas razoáveis de recursos, custos e prazos para cada resultado esperado em um projeto. De posse da proposta de projeto, um "plano de projeto"deve ser elaborado de forma a customizar o processo a ser utilizado no desenvolvimento desses resultados. À medida que o projeto progride, o plano de projeto deve ser detalhado e atualizado regularmente. Pelo menos ao final de cada uma das fases do desenvolvimento, o planejamento como um todo deve ser revisto e o planejamento da etapa seguinte deve ser detalhado. O planejamento e o acompanhamento do progresso fazem parte das atividades de gerência de projeto.

- **Concepção:** Nesta fase, os processos de levantamento e análise de requisitos são intensificados. O escopo do projeto deve ser refinado e os requisitos detalhados. A principal questão é entender "o que"deverá ser construído. Para entender a natureza dos produtos que serão construídos, os coordenadores de projetos e desenvolvedores têm de compreender o domínio do problema, bem como as funcionalidades e os comportamentos esperados. Uma vez capturados os requisitos dos produtos, estes devem ser modelados, avaliados e documentados. Uma parte vital desta fase é a construção do escopo negativo, ou seja, o que os produtos não serão capazes de fazer.

- **Planejamento:** Esta fase implica em definir "como"cada resultado esperado do projeto será construído. Quando o resultado esperado é um software/modelo, seu projeto arquitetural deve ser definido, além do projeto das estruturas de dados e algoritmos que serão empregados para sua construção. A arquitetura deve descrever a estrutura de nível mais alto da aplicação, identificar seus principais módulos e como eles se relacionam. A

seguir, é preciso detalhar a hierarquia de classes e objetos para cada um desses módulos. Os módulos devem ser sucessivamente refinados em níveis maiores de detalhamento, até que possam ser codificados e testados. Os testes unitários e de integração também devem ser detalhados ao máximo durante essa fase.

- **Construção:** Este é o momento da implementação dos resultados esperados em uma forma passível de execução pela máquina, ou seja, o código fonte de cada módulo de software/modelo planejado é produzido. Partindo-se do pressuposto de que construir algo é construir algo que funcione, durante essa fase, os testes unitários de cada módulo de software/modelo são implementados e utilizados para avaliá-los. O objetivo é homologar cada módulo unitário.

- **Testes:** Esta fase inclui os testes de integração e os testes para avaliação de requisitos não funcionais como desempenho, segurança, disponibilidade e usabilidade. Os testes de integração visam garantir o correto funcionamento coordenado de todos módulos desenvolvidos. Todos esses teste precisam ser detalhados, implementados e analisados nessa fase. O objetivo é homologar a arquitetura de software integrada e garantir que ela atenda aos requisitos não funcionais.

- **Implantação:** Uma vez testados, os resultados do projeto devem ser empacotados e distribuídos. Para isso, os testes de empacotamento e da mídia de distribuição devem ser construídos e avaliados nessa fase. Muitas vezes, os resultados de um projeto precisam ser colocados em produção no ambiente do cliente. Geralmente, é necessário treinar os usuários, configurar o ambiente de produção e realizar a operação assistida desses produtos junto com o usuário. O propósito desta fase é garantir que os produtos satisfaçam os requisitos dos usuários por meio de sua maturação. Isto é feito conduzindo-se testes de aceitação. Quando um produto tiver demonstrado prover as capacidades requeridas, ele pode deve ser aceito pelo cliente e o projeto de desenvolvimento chega ao fim. Então, um projeto de Operação e Manutenção pode ser iniciado.

3.5 Gestão de projetos do TerraLAB

As atividades de cada fase do projeto são agrupadas nas disciplinas enumeradas no processo de desenvolvimento de software RUP: gestão de projetos, gestão de configuração, modelagem do negócio, planejamento, construção, etc.

A gestão de projetos no TerraLAB se orienta à lidar com equipes multidisciplinares visando o desenvolvimento de inovações, desenvolver e utilizar habilidades que combinem arte (criação) e ciência (métodos) e, finalmente, surpreender os clientes que podem ser institutos de pesquisa, agências de fomento à pesquisa ou empresas privadas. Para isso, manter as equipes motivadas e trabalhando de forma produtiva e colaborativa, além de administrar da melhor maneira os recursos de tempo, pessoal e financeiro são atividades essenciais.

Durante as atividades de análise de negócio, o gerente de projeto e o analista de negócio baseiam-se no documento de proposta de projeto para elaborar o documento de plano de projeto. Este último documento define "como" o projeto será executado, monitorado, controlado e encerrado. As principais questão que devem ser respondidas são: O que deve ser feito no futuro para atingir os objetivos do projeto? Como vai ser feito? Quem o vai fazer? Quando estará feito? No TerraLAB, essas questões são parcialmente respondidas por processos de gestão de projetos e de desenvolvimento de software/modelos padronizados. Desta forma, o documento de plano de projeto pode ser simplificado, bastando que ele customize os processos pré-existentes, definindo em que nível eles serão implementados para cada entregável a ser produzido. Além disso, ele define o cronograma detalhado do projeto e identifica os principais marcos de projeto (*milestones*). O cronograma detalhado informa as tarefas que deverão ser realizadas para o desenvolvimento de cada entregável, o prazo estimado de cada tarefa, os recursos necessários e o responsável por sua execução. Parte-se do pressuposto de que sem um entendimento completo do problema a ser tratado e sem um planejamento bem feito em mãos, o gerente de projetos, o gerente de fábrica e suas equipes não saberão onde e quando desejam chegar, nem do que necessitam para isso.

Uma vez detalhado, o cronograma de projeto é cadastrado no sistema de gestão de projetos NetProject que permite o acompanhamento on-line dos projetos em andamento por meio de: Semáforos que sinalizam o estado de cada entregável (atrasado, em andamento, concluído), relatórios que indicam o percentual concluído de cada entregável e atividades planejadas, gráficos de Gantt que contrastam prazos estimados e executados, gráficos de barra que mostram as horas trabalhadas por cada membro de equipe, gráficos que mostram o percentual de alocação de cada recurso e quanto foi efetivamente consumido. Cotidianamente, ao iniciar sua jornada de trabalho, onde quer que o membro de equipe esteja localizado, ele deve acessar o sistema NetProject via Internet, selecionar o projeto em que deseja trabalhar naquele momento e registrar entrada na tarefa que ele deseja executar. Ao final da jornada de trabalho, ele deve efetuar a saída da tarefa em que estava registrado e informar o percentual de completude dessa

Ações		#	Nome da Atividade	Início Planejado	Término Planejado	Duração Planejada (dias)	Status	% Conclusão	Responsável Execução	Responsável Aprovação
		0	MeuProjeto2011/2	09/11/2011 08:00	26/01/2012 17:00	57	Em execução	22,09	igor.muzetti	tiago.carneiro
		1	Entregável 1	09/11/2011 08:00	16/12/2011 17:00	28	Em execução	28,42	tiago.carneiro	tiago.carneiro
		2	Fase 1 - Análise de Negócio	09/11/2011 08:00	15/11/2011 17:00	5	Concluído	100,00	tiago.carneiro	tiago.carneiro
		3	Atividade 1 - Modelar Negócio	09/11/2011 08:00	15/11/2011 17:00	5	Concluído	100,00	tiago.carneiro	tiago.carneiro
		4	Atividade 2 - Levantar e Analisar os Requisitos	09/11/2011 08:00	15/11/2011 17:00	5	Concluído	100,00	xaviererika	tiago.carneiro
		5	Atividade 3 - Projetar	09/11/2011 08:00	15/11/2011 17:00	5	Concluído	100,00	joubert.lima	tiago.carneiro
		6	Atividade 4 - Implementar	09/11/2011 08:00	15/11/2011 17:00	5	Concluído	100,00	bastosaugusto	tiago.carneiro
		7	Atividade 5 - Testar	09/11/2011 08:00	15/11/2011 17:00	5	Concluído	100,00	xaviererika	tiago.carneiro
		8	Atividade 6 - Gerenciar Configuração e Mudança	09/11/2011 08:00	15/11/2011 17:00	5	Concluído	100,00	rreisp	tiago.carneiro
		9	Atividade 7 - Gerenciar Projeto	09/11/2011 08:00	15/11/2011 17:00	5	Concluído	100,00	igor.muzetti	tiago.carneiro
		10	Atividade 8 - Gerenciar Ambiente de Desenvolvimento	09/11/2011 08:00	15/11/2011 17:00	5	Concluído	0,00	igor.muzetti	tiago.carneiro
		11	Fase 2 - Concepção	09/11/2011 08:00	22/11/2011 17:00	10	Em execução	0,81	tiago.carneiro	tiago.carneiro
		12	Atividade 1 - Modelar Negócio	09/11/2011 08:00	22/11/2011 17:00	10	Em execução	1,00	tiago.carneiro	tiago.carneiro
		13	Atividade 2 - Levantar e Analisar os Requisitos	09/11/2011 08:00	22/11/2011 17:00	10	Em execução	1,00	tiago.carneiro	tiago.carneiro
		14	Atividade 3 - Projetar	09/11/2011 08:00	22/11/2011 17:00	10	Em execução	1,00	joubert.lima	tiago.carneiro
		15	Atividade 4 - Implementar	09/11/2011 08:00	09/11/2011 17:00	1	Em execução	1,00	rreisp	tiago.carneiro
		16	Atividade 5 - Testar	09/11/2011 08:00	09/11/2011 17:00	1	Em execução	1,00	igor.muzetti	tiago.carneiro
		17	Atividade 6 - Gerenciar Configuração e Mudança	09/11/2011 08:00	09/11/2011 17:00	1	Em execução	1,00	rreisp	tiago.carneiro
		18	Atividade 7 - Gerenciar Projeto	09/11/2011 08:00	22/11/2011 17:00	10	Em execução	1,00	joubert.lima	tiago.carneiro
		19	Atividade 8 - Gerenciar Ambiente de Desenvolvimento	09/11/2011 08:00	22/11/2011 17:00	10	Em execução	0,00	igor.muzetti	tiago.carneiro
		20	Fase 3 - Planejamento	09/11/2011 08:00	25/11/2011 17:00	13	Aguardando Execução	0,00	tiago.carneiro	tiago.carneiro
		29	Fase 4 - Construção	09/11/2011 08:00	02/12/2011 17:00	18	Aguardando Execução	0,00	tiago.carneiro	igor.muzetti
		38	Fase 5 - Testes	09/11/2011 08:00	09/12/2011 17:00	23	Aguardando Execução	0,00	tiago.carneiro	tiago.carneiro
		47	Fase 6 - Implantação	09/11/2011 08:00	16/12/2011 17:00	28	Aguardando Execução	0,00	tiago.carneiro	tiago.carneiro
		56	Entregável 2	16/12/2011 00:00	26/01/2012 17:00	30	Aguardando Execução	0,00	igor.muzetti	tiago.carneiro

Figura 3.3: **Modelo de Cronograma TerraLAB**

tarefa. Todo o período de tempo decorrido é automaticamente registrado como horas trabalhadas para o membro de equipe e para o cronograma executado do projeto. Desta maneira, cada aspecto relevante de um projeto pode ser gerido com base em indicadores quantitativos atualizados diariamente. Para que a gestão de projetos funcione com eficácia, é essencial que essa cultura de uso de ferramenta de gestão de projetos seja entendida e empregada por todas as equipes de projeto.

O modelo de cronogramas de projeto do TerraLAB foi definido e é apresentado na figura 3.3. Nele, um projeto é primeiramente decomposto em entregáveis que, por sua vez pode ser desmembrado em sub-entregáveis. Então, cada entregável é decomposto nas fases que formam o ciclo de vida do projeto e a cada fase, as atividades de desenvolvimento de software são desempenhadas. Essa estrutura de cronograma tem como foco o acompanhamento individual de cada entregável de um projeto. A vantagem identifica nesta forma de organização são: (i) Permitir que cada entregável evolua de forma independente por diferentes estágios de desenvolvimento; (ii) Fácil identificação da fase de desenvolvimento de cada entregável; (iii) Fácil aferição do porcentual de completude de cada entregável; (iv) Organiza os artefatos dos processos produtivos por fase forma que, a cada fase, artefatos diferentes recebem o foco da equipe; (v) Facilita o mapeamento de dependência entre entregáveis. A principal desvantagem é para um relato do estágio de desenvolvimento do projeto como um todo é necessária a análise em separado sobre o estágio de desenvolvimento de cada e entregável. Contudo, a flexibilidade

da estrutura sobrepujar essa desvantagem.

É importante dizer que as atividades de desenvolvimento executadas a cada fase são executadas segundo os processos padronizados e descritos na próxima seção. Estes processos determinam a sequência de passos que deve ser seguida por um membro de equipe a fim de produzir um determinado tipo de entregável. Além disso, os processos estabelecem quais artefatos devem ser confeccionados a cada passo e quais são os modelos padronizados desses artefatos.

No que diz respeito à mensuração da qualidade dos entregáveis produzidos, cada coordenador de projeto deve garantir com a robustez, usabilidade e desempenho dos produtos desenvolvidos. Toda a equipe deve ter consciência de que não adianta desenvolver produtos que só funcionem em condições especiais, com interfaces feias e ilegíveis, muito lentos ou que consumam memória exageradamente. Para evitar falhas nas funcionalidades dos produtos, os casos de teste não podem ser insuficientes, ingênuos ou executados sem devida atenção, de forma displicente e imprudente. O gerente de fábrica deve cobrar de cada coordenador de equipe que todos esses requisitos sejam satisfeitos, além de avaliar a qualidade dos artefatos gerados pelas equipes.

3.6 Processo de desenvolvimento de software/modelo BOPE

Qualquer processo de desenvolvimento de software poderia ser associado à metodologia de gestão projetos proposta pelo PMI para a execução eficaz de projetos de software. Portanto, não é necessário que um laboratório de pesquisa adote o processo BOPE para utilizar a mesma metodologia de gestão de projetos do TerraLAB e integrar-se administrativamente a ele. Nem mesmo, demanda-se que adoção de uma mesma estrutura organizacional. Contudo, para projetos que precisam ser comparados em termos da viabilidade de suas propostas técnicas, em termos da quantidade e qualidade dos resultados produzidos ou, ainda, em termos da produtividade de suas equipes é importante o uso de artefatos comuns para sua proposição, monitoramento e avaliação.

O contexto acadêmico no qual um laboratório de pesquisa em computação se insere nas universidades brasileiras traz consigo desafios que não existem em outros contextos. Para contornar esses desafios, o processo de desenvolvimento de software/modelo proposto neste trabalho combina o formalismo e os padrões do processo RUP com a flexibilidade e reatividade às mudanças do processo Scrum. No processo BOPE (Boosted Object-oriented software Process Engineering), para cada fase do ciclo de vida de um projeto, existe um conjunto de processos padrões que devem ser seguidos para que os resultados esperados sejam produzidos. Cada processo é descrito por um fluxograma que mostra como sequências de atividades precisam ser realizadas a fim conferir qualidade a esses resultados. Cada atividade, por sua vez, dá origem a um fluxo de tarefas que transformam artefatos padronizados de entrada em artefatos padronizados de saída. Cada uma dessas tarefas possui um único responsável por sua realização. A figura 3.4 mostra como o processo de vendas do TerraLAB deve ser realizado. A figura 3.5 mostra o fluxo de tarefas que deve ser realizado para completar a atividade "reune com cliente". Para cada uma das tarefas é definido o papel do responsável por sua execução, os artefatos de entrada e os artefatos de saída.

O BOPE é mantido em constante evolução. Uma equipe de melhoria de processos busca continuamente torná-lo mais eficaz, isto é, menos burocrático e capaz de produzir melhores resultados. À medida que essa equipe entende melhor o nicho em que a fábrica de software atua e conhece mais a respeito dos tipos de produtos da fábrica, ela torna-se mais capaz em projetar os processos produtivos desta fábrica.

No entanto, segundo o BOPE, com a exceção dos processos de gestão de projetos, os nível de implementação dos processos da fábrica pode ser customizado pelo analista de negócios no momento da elaboração do documento de plano de projeto. Desta maneira, à medida que ele julga uma equipe de projeto mais experiente e capaz de produzir bons resultados com menos burocracia, ele pode permitir que artefatos menos custosos e fora do padrão sejam produzidos. Para isso, o analista de negócio precisa estabelecer um contrato entre ele, o gerente de fábrica e o gerente de projetos. O Anexo I apresenta alguns fluxos de processos e os fluxos de atividade

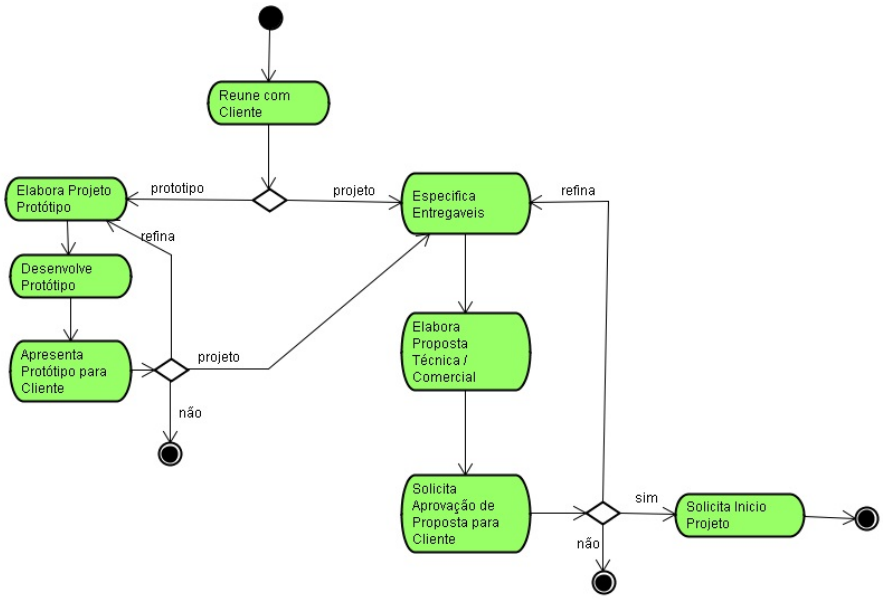


Figura 3.4: **Processo de vendas do TerraLAB, realizado para captação de novos projetos.**

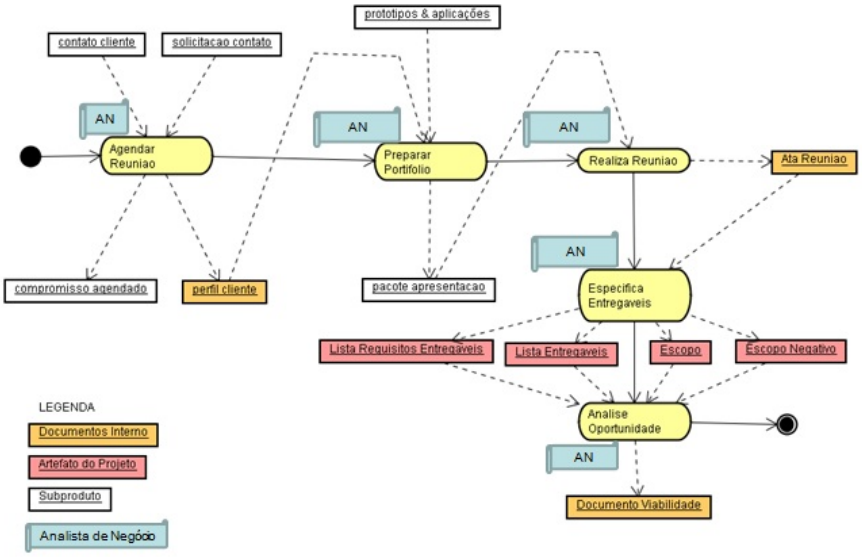


Figura 3.5: **Atividade reunir com cliente do processo BOPE.**

atualmente definidos para o BOPE. Outra condição que pode justificar a customização de um processo é o nível de dedicação do analista de negócio ao projeto. Nos projetos de pesquisa, um pesquisador pode comprometer-se pessoalmente com a qualidade dos resultados esperados para um projeto e, então, dedicar seu tempo a essa atividade. Contudo, à medida que ele o faz, sua capacidade de assumir novos projetos diminui.

Nestas condições, o processo BOPE se assemelha ao processo Scrum, as interações devem ocorrer em intervalos mais curtos, conduzidos por comunicações face-a-face, envolvendo menos documentação, mais atividades orientadas a código e mais colaboração com um representante do cliente que, neste caso, é um papel desempenhado pelo próprio pesquisador. No entanto, é importante refletir que em ultima instância o cliente de um projeto de pesquisa é quem o está financiando e deseja receber os entregáveis. O projeto com um todo é entendido com o product backlog, cada conjunto de artefatos que precisas ser produzidos até um determinado marco de projeto é visto com um sprint backlog. Um sprint em realizar todas as atividades de desenvolvimento até que todos esses artefatos sejam produzidos. Devido a baixa dedicação e baixo comprometimento dos membros de equipe, sprints têm duração mínima de 1 a 3 meses, pois reuniões que permitem o acompanhamento e gerência de riscos do projeto somente podem ser realizadas em ciclos ocorrem de 1 a 7 dias, como ilustra a figura 3.6.



Figura 3.6: **Semelhança entre o processo de desenvolvimento de software BOPE e o processo Scrum.**

A evolução cíclica e incremental do projeto permite revisões periódicas. A cada vez que um ciclo completo de atividades de desenvolvimento acontece, ou seja, a cada 1 ou 3 meses, em geral o projeto avança uma fase em seu estágio de desenvolvimento e as atividades da nova fase são então detalhadas em seu plano. A cada reunião de avaliação e planejamento, que acontece de 1 a 7 dias, as atividades previstas no plano de projeto são transformadas em uma sequência de solicitações de mudança registradas no sistema de controle de mudanças. Estas solicitações são decididas em conjunto pela equipe de projeto e representam os passos que devem ser seguidos para que a atividades seja realizada com sucesso. Desta maneira, o processo BOPE integra os processos de gestão de projetos e de controle de mudanças.

3.7 Artefatos de Software TerraLAB

Para cada tipo de produto desenvolvido pelo TerraLAB foi definido um conjunto de artefatos padronizados que precisam ser produzidos e detalhados ao longo do ciclo de vida dos projetos. Atualmente, o TerraLAB executa projetos de pesquisa no qual monografias e artigos científicos são os principais resultados almejados. Ele também executa projetos de desenvolvimento e inovação onde os principais resultados são software e modelos de simulação. Para cada um desses tipos de produto, um conjunto diferente de artefatos deve ser produzido. A descrição de cada um desses artefatos foge ao escopo deste trabalho. O Anexo II apresenta padrões de alguns artefatos dos projetos de software executados pelo TerraLAB. Eles são apresentados na forma de um modelo padronizado e comentado que precisa somente ser customizado pelos membros de equipe. A figura 3.7 mostra os artefatos organizados por tipo de projetos e fases do ciclo de vida dos projetos.

Fases	Análise de Negócio	Concepção	Planejamento	Construção	Testes	Implantação
Artefatos (Software)	Proposta de Projeto, Plano de Projeto, Documento de Escopo	Documento de Requisitos, Diagrama de Caso de Uso	Diagrama de Classes, Diagrama de Sequência, Plano de Testes Unitários	Código Fonte, Testes Unitários, Relatório de Testes Unitários	Plano de Testes de Integração, Relatório de Resultados dos Testes de Integração	Preparação do ambiente do usuário, execução do entregável no ambiente do cliente
Artefatos (Modelo)	Proposta de Projeto, Plano de Projeto, Documento de Escopo	Documento de Revisão de Trabalhos Correlatos, Documento de Revisão de Conceitos Básicos, Documento de Requisitos, Workshop com Especialistas, Experimentos Piloto, Documento de Escopo de Cenário	Diagrama de Fluxo de Energia, Projeto de Trabalho de Campo, Projeto de Experimentos, Projeto de BD, Projeto de Cenário	Implementação, Verificação do Modelo, Análise de Sensibilidade, Calibração, Implementação do Banco de Dados, Implementação de Cenários	Validação, Simulação de Cenários	Relatório, Artigo Científico, Dsiponibilização do Modelo e Documentos na Wiki do TerraLAB

Figura 3.7: **Artefatos padronizados do TerraLAB.**

3.8 Ferramentas CASE utilizadas pelo TerraLAB

Os requisitos determinados por cada processo de desenvolvimento de software do TerraLAB foram considerados critérios cruciais para escolha das ferramentas CASE de apoio aos processos. Por exemplo, o processo de gestão de projetos exige conformidade com metodologia proposta pelo PMI. Sempre que possível as ferramentas livres foram privilegiadas. No entanto, a parceria com a empresa de gestão de projeto NetProject, desenvolvedora do sistema de gestão de projeto de mesmo nome, permitiu o incorporação da metodologia PMI em seus processos e uso irrestrito desse sistema para a gestão de projetos do TerraLAB.

O sistema de controle de mudanças TRAC é utilizado para realizar o controle de mudança nos projetos de desenvolvimento de software/modelo do TerraLAB. Além do controle de mudanças, o sistema TRAC também é utilizado para organizar as tarefas cotidianas do laboratório, isto é, tarefas operacionais que não foram previstas nos cronogramas de cada projeto, mas que beneficiam a todos os projetos. Por exemplo, comprar papel para impressora, corrigir bugs, fazer *backup*s, configuração de alguma estação de trabalho, etc. Desta maneira, a manutenção do próprio laboratório é vista como um projeto que tem início no primeiro do ano e termina no ultimo dia.

É preciso esclarecer que solicitações de mudanças são registradas no TRAC a todo o momento e que, por isso, não é possível prever quantas solicitações de mudança serão atendidas em um projeto. A conclusão de 50% dessas solicitações não significa que 50% de um projeto realizado. No entanto, quando os membros da equipe dizem que 50% de todas as atividades previstas no cronograma do NetProject foram completadas, então temos certeza que 50% do projeto foi concluído. Logo, enquanto o NetProject permite controlar quantitativamente cronogramas, recursos e resultados dos projetos, o TRAC ajuda a organizar as tarefas cotidianas e a priorizá-las. Caso não houvesse imprevistos ou tipos de produtos para os quais não existe processo definido (caso recorrente em projetos de pesquisa), cada solicitação de mudança registrada no TRAC corresponderia a uma tarefa para a transformação de artefatos de entrada em artefatos de saída, conforme previsto em um processo padronizado.

No TerraLAB, a ferramenta SVN é utiliza para o controle de versões de software/modelo. O ferramenta de WIKI denominada DokuWiki é utilizada para a documentação colaborativa dos projetos.

3.9 Infraestrutura de hardware e software no TerraLAB

A implementação da fábrica requer considerações especiais quanto aos equipamentos alocados para sua operação. Devem existir estações de trabalho para o desenvolvimento dos produtos da fábrica, servidores para execução das ferramentas CASE de apoio a seus processos produtivos e servidores para administração de usuários, administração de recursos compartilhado (disco, impressora, etc) e execução do sistema de *backup*. O TerraLAB atualmente conta com 7 estações de trabalho com dois sistemas operacionais instalados, permitindo o desenvolvimento de aplicações para Windows e Linux. Um único servidor interno na rede militarizada executa os serviços de autenticação e de sistema de arquivos (Samba), executa o sistema de gestão de projetos (NetProject), o sistema de controle de mudanças (TRAC) e sistema de controle de versão (SVN). Outro servidor, localizado na porção desmilitarizada da rede TerraLAB, executa os serviços de firewall, tradução de endereços (NAT) e o sistema de WIKI (DokuWiki). O serviço de *backup* (BACULA) é executado de maneira espelhada nesses dois servidores, cada um possui um disco rígido dedicado exclusivamente a essa atividade. Todos os finais de semana, um *backup* incremental é realizado e, uma vez por mês, um *backup* total é automaticamente realizado.

O TerraLAB possui dois equipamentos de alto desempenho que atuam como servidores de simulação. Além disso, muitos experimentos são realizados em um supercomputador compartilhado com outros laboratórios do Departamento de Computação da UFOP.

3.10 Experimentos: Implantação do processo BOPE no TerraLAB

Os projetos SIGHabitar e o TerraME foram escolhidos para implantação do processo de desenvolvimento BOPE em conjunto com a gestão de projeto da fábrica PMI. O SIGHabitar tem pouca complexidade científica e muita demanda por recursos, uma equipe formada por 15 pessoas, em geral estagiários, alunos de mestrado ou iniciação científica. O TerraME é muito complexo cientificamente e vem sendo mantido por uma equipe pequena e altamente qualificada. Esses projetos pilotos foram acompanhados ao longo do tempo. À medida que o processo BOPE era implantado, indicadores de produtividade foram coletados, como é possível observar na figura 1.1, e foram tomadas medidas corretivas a fim de melhorar o BOPE, tornando-o definido, monitorado, planejado e controlado.

Capítulo 4

Resultados

Neste trabalho, o laboratório de pesquisa, desenvolvimento e inovação em computação TerraLAB foi abordado como uma fábrica de software, seu nicho de atuação e estrutura organizacional foram formalmente definidos. Os deveres e direitos de cada papel presente em sua estrutura organizacional foram enumerados.

A metodologia de gestão de projeto PMI foi implantada em seu ambiente produtivo e, atualmente, todos os projetos em execução atendem aos requisitos mínimos sugeridos no PMBOK. Para isso, um ciclo de vida comum a todos os projetos foi padronizado, diferentes ferramentas de apoio a gestão de projetos foram avaliadas e, finalmente, a cultura dos membros de equipe foi incrementalmente alterada para que fizessem uso da metodologia PMI e do sistema NetProject.

Para contornar os desafios encontrados por um ambiente de produção de software inserido no ambiente acadêmico das universidades públicas brasileiras, o processo de desenvolvimento de software BOPE foi formalmente definido e seus artefatos padronizados. Posteriormente, o processo BOPE foi implantado em dois projetos piloto: TerraME e SIGHabitar. Durante, os ciclos de desenvolvimento desses projetos o processo BOPE sofreu diversas melhorias que resultaram em práticas menos burocráticas e artefatos menos complexos e mais eficazes. Os fluxos que representam os processos que formam o BOPE podem ser encontrados no Anexo I que acompanha este texto. Para suporte e automação dos processos, diversos sistemas de controle de versão e de controle de mudanças foram avaliados. Os sistemas SVN e TRAC foram selecionados e implantados em todos os projetos em andamento no laboratório. O sistema DokuWiki tem sido utilizado com êxito para documentar todos projetos.

A gerência de projetos e de processos do TerraLAB buscou assegurar que os artefatos e a execução dos processos estejam em conformidade com os planos, prazos e recursos pré-definidos. A definição do processo de garantia da qualidade consistiu em criar atividades de auditoria, definir políticas, periodicidade e checklists de avaliação. As auditorias de qualidade são reuniões com o analista de negócio da fábrica e/ou com um representante do cliente.

Juntas estas atividades são uma forma da fábrica garantir que os projetos estejam seguindo os processos definidos, gerando os artefatos esperados e dentro dos prazos estabelecidos.

A metodologia de gestão de projeto tem permitido o efetivo acompanhamento e avaliação dos projetos, além da comparação dos resultados obtidos por diferentes equipes. A maioria dos riscos passou a ser tratada de maneira preventiva. A cada novo projeto, os prazos e recursos estimados se aproximam dos prazos e recursos consumidos. Os clientes constantemente elogiam a visibilidade que os relatórios de acompanhamento do projeto lhes oferecem. Os artefatos formais têm permitido a substituição de membros de equipe sem que os prejuízos trazidos por essa mudança inviabilizem os projetos. O uso de processos formais e artefatos padronizados têm permitido que membros de equipe com pouquíssima experiência produzam resultados de boa qualidade.

Capítulo 5

Discussão e Trabalhos Futuros

O gerenciamento de projetos e o processo de desenvolvimento de software BOPE formam um modelo de desenvolvimento de software/modelo que serve como um guia para a fábrica TerraLAB. Entretanto, apesar de mais de um ano realizando estudos para a definição e implantação desse modelo, considera-se que ele ainda está em processo de aprimoramento, devendo sofrer revisões periódicas. Desta maneira, a fabrica TerraLAB deve continuamente buscar a melhoria de seus processos produtivos e, futuramente, ser reconhecida em um nível de maturidade avaliado pelo MPS.BR.

Entretanto, tentar atropelar e forçar os limites da fábrica pode trazer grandes prejuízos a sua produtividade. Desta forma, a implantação desse modelo deve acontecer de maneira incremental e deve permitir que a maturidade dos processos da fábrica evolua passo a passo, fundamentada em definições claras para os membros de equipe, convencido da importância dos processos à medida que eles observam ganhos nos resultados que produzem. Manter as equipes sempre motivadas é vital para o sucesso da implantação desse modelo.

O estabelecimento e manutenção da integridade dos artefatos dos projetos e utilização rotineira do sistema TRAC integrado ao SVN, estabeleceu uma sólida base o gerenciamento de configuração. O próximo passo é redefinir as estruturas de repositório, padrões de nomenclatura, registros de linhas de base, regras de integração de arquivos ao repositório e checklists para auditoria do sistema de configuração para alcançar o objetivo de manter o repositório íntegro e gerar informações para que se possa obter a rastreabilidade entre requisitos e código fonte e vice-versa.

O uso de frameworks e bibliotecas facilitaram a reutilização de componentes, ajudaram a aumentar a produtividade da equipe e diminuem os custos associados aos projetos. Treinamentos de todos os membros de equipe sobre o processo, bem como o uso de tecnologias padronizada no ambiente do laboratório ajudaram a criar uma comunicação uniforme entre as equipe, a coordenar suas atividades e, por conseguinte, aumentaram a motivação de seus integrantes. A publicação de uma wiki de desenvolvimento facilitou a comunicação com o

cliente e com os próprios colaboradores, pois ela disponibiliza informações gerais, tutoriais, padrões de artefatos, progresso de desenvolvimento, metas dos projetos e informações sobre a equipe.

Considera-se que o ganho de conhecimento nos experimentos realizados e a aplicação do BOPE no TerraLAB foram satisfatórios. Contudo, há ainda muito trabalho a ser feito para que a fábrica atinja ótimo nível de maturidade organizacional, principalmente no que diz respeito ao gerenciamento de projeto com produtividade e qualidade aceitáveis. O propósito em atingir um nível ótimo consiste em estabelecer e manter planos que definam as atividades, recursos e responsabilidades do projeto, bem como prover informações sobre o andamento do projeto que permitam a realização de correções quando houver desvios significativos no desempenho do mesmo. Os experimentos mostraram que a implantação de um processo de desenvolvimento de software é custosa e representa um grande investimento inicial de esforço dos envolvidos, porém o TerraLAB tende a obter maior sucesso e retorno deste investimento pois o processo foi elaborado e construído dentro dele próprio. O BOPE não se impõe às pessoas, ele foi construído junto com elas.

Neste trabalho, foi possível verificar que para que o processo de desenvolvimento adotado em laboratórios de pesquisa em computação situados em instituições de ensino e pesquisa alcance produtividade e qualidade satisfatórias, ele deve incorporar características encontradas em metodologias ágeis, como minimização e simplificação das tarefas e artefatos, e grande comunicação entre a equipe e o cliente. Apesar disso, características encontradas em processos mais maduros como o RUP não podem ser ignorados, como o auxílio ao processo de venda, a garantia da qualidade e a redução dos riscos no início do projeto através da definição e uso de um modelo explícito.

Finalmente, observa-se que o modelo de gerenciamento de projetos, o processo BOPE e as ferramentas de apoio utilizados pelo TerraLAB, podem serem usados como referência para ambientes de produção de software que compartilham desafios semelhantes aos enfrentados pelo TerraLAB.

Referências Bibliográficas

Craig Larman, V. R. B. (2003). Iterative and incremental development: A brief history. *IEEE Computer Society Press Los Alamitos*, 36(6):47–56.

Cusumano, M. A. (1991). *Japan's Software Factories*. Oxford University Press.

Edes Garcia da Costa Filho, Rosângela Aparecida Delloso Penteado, J. C. A. S. R. T. V. B. (2005). Padrões e métodos Ágeis: agilidade no processo de desenvolvimento de software. *5ª Conferência Latino-Americana em Linguagens de Padrões para Programação*, 1:172–185.

Fabiane Barreto Vavassori, Everton Wilson de Souza, J. C. F. (2001). Ferramenta case para gerenciamento de projetos e métricas de software. *XV Simpósio Brasileiro de Engenharia de Software*, pp. 362 – 367.

Husu, M. (2006). Software factories.

IDEF0 (2011). Integration definition language. http://www.idef.com/.

IEC 12207 (1998). Abnt - associaÇÃo brasileira de normas tÉcnicas - tecnologia da informação - processos de ciclo de vida de software.

IEEE (1987). Guide to software configuration management. Institute of Electrical and Electronics Engineers.

ISO 9000 (2005). Abnt - associaÇÃo brasileira de normas tÉcnicas - sistemas de gestão da qualidade - fundamentos e vocabulário.

Ivar Jacobson, Grady Booch, J. R. (1999). *The Unified Software Development Process*. Addison Wesley Longman.

Kent Swanson, Dave McComb, J. S. D. M. (1991). The application software factory: applying total quality techniques to systems development. *MIS Quarterly*, 15(4):567–579.

Larman, G. (2007). *Utilizando UML e padrões*. Bookman, Brasil, 3 edição.

Leandro Alves Patah, M. M. d. C. (2002). Estruturas de gerenciamento de projetos e competências em equipes de projetos. *XXII Encontro Nacional de Engenharia de Produção*.

Luzia Nomura, Mauro de Mesquita Spínola, A. C. T. O. K. H. (2007). A model for defining software factory processes. *19th International Conference on Production Research.*

Manifesto for Agile Software Development (2001). Manifesto for agile software development. http://agilemanifesto.org/.

Mark C. Paulk (1993). Capability maturity model for software.

Nokes, S. (2007). *The Definitive Guide to Project Management: The Fast Track to Getting.* Pearson Education.

Phillips, J. (2003). *Gerência de projetos de tecnologia da informação.* Elsevier, Brasil, 10 edição.

PMBOOK, P. M. I. (2000). *A Guide to the Project Management Body of Knowledge.* PMBOK® Guide, Pennsylvania-USA, 3 edição.

Pressman, R. S. (2006). *Engenharia de Software.* McGraw-Hill, Brasil, 6 edição.

RUP (2006). Rup - ibm rational unified process. http://www-306.ibm.com/software/rational/.

Schwaber, K. (2004). *Agile Project Management with Scrum.* Microsoft Press.

Sircar, S.; Nerur, S. P. e Mahapatra, R. (2001). Revolution or evolution? a comparison of object-oriented and structured systems development methods. *MIS Quarterly*, 25:457–471.

Softex - MPS.BR (2011). Melhoria de processo do software brasileiro. http://www.softex.br/mpsbr/.

Tim Sloane (2003). Business model collaborations: Pushing software revolution. http://www.softwaremag.com/.

ANEXOS

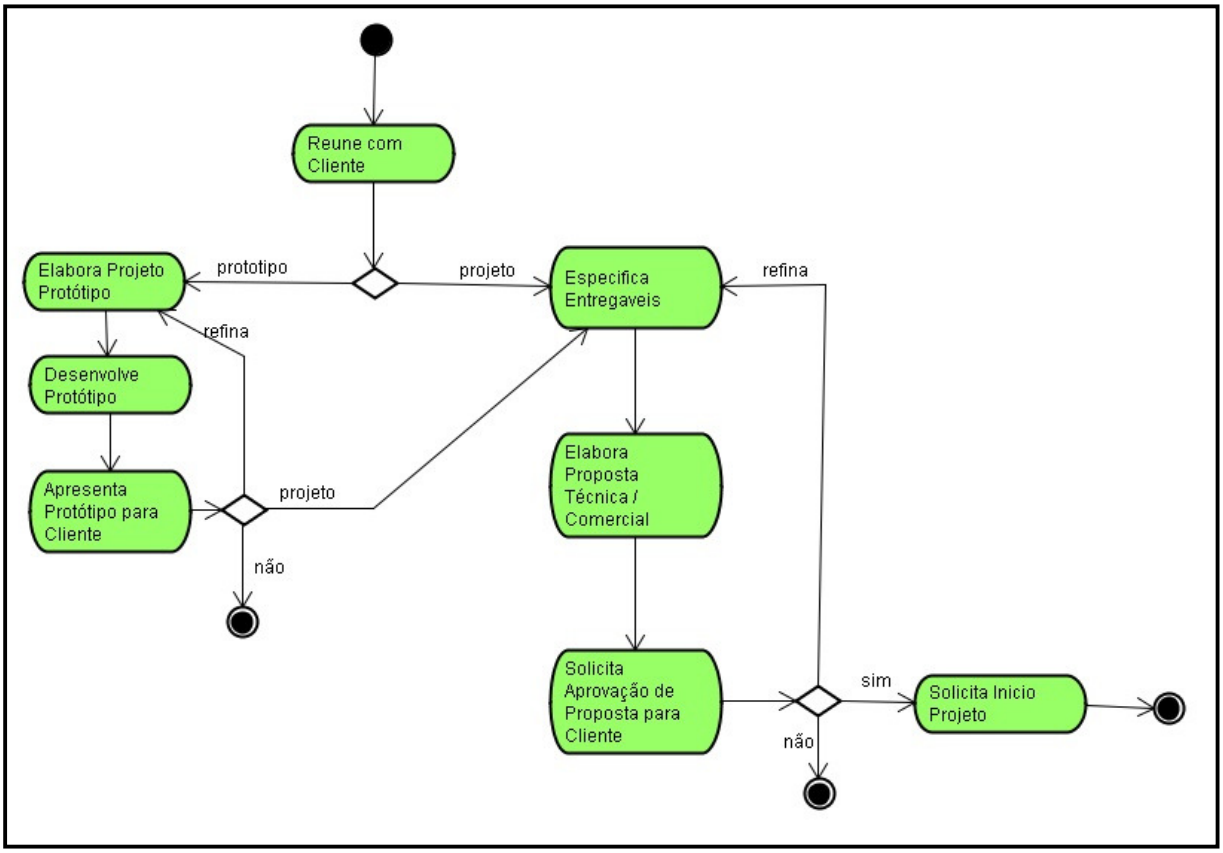
Anexo I – Processo de Desenvolvimento de Software TerraLAB.

Anexo II - Artefatos de Software TerraLAB (Documento de Requisitos, Documento de Caso de Uso, Plano de testes, Relatório de resultados dos testes, Relatórios de Acompanhamento).

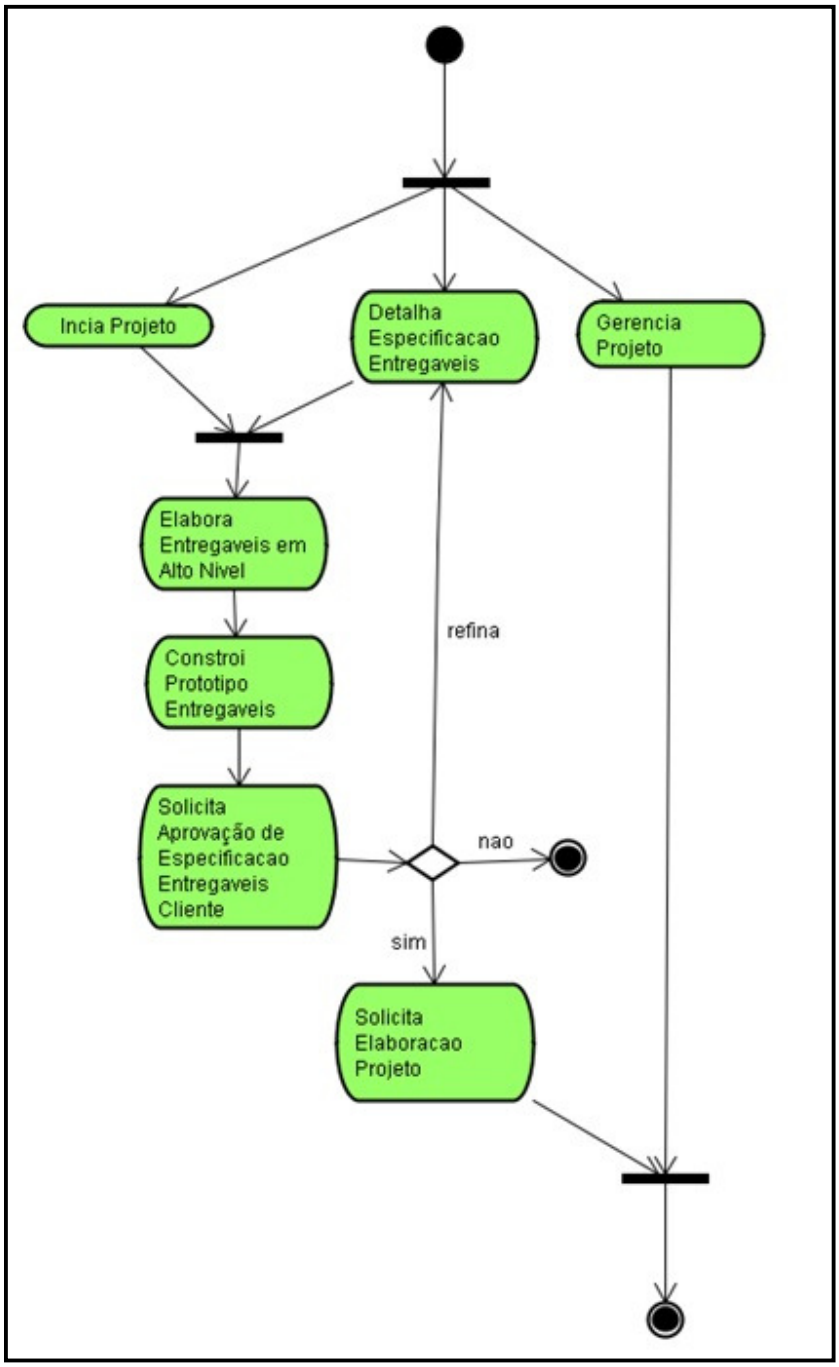
ANEXO I

FASES DO CICLO DE VIDA DE SOFTWARE

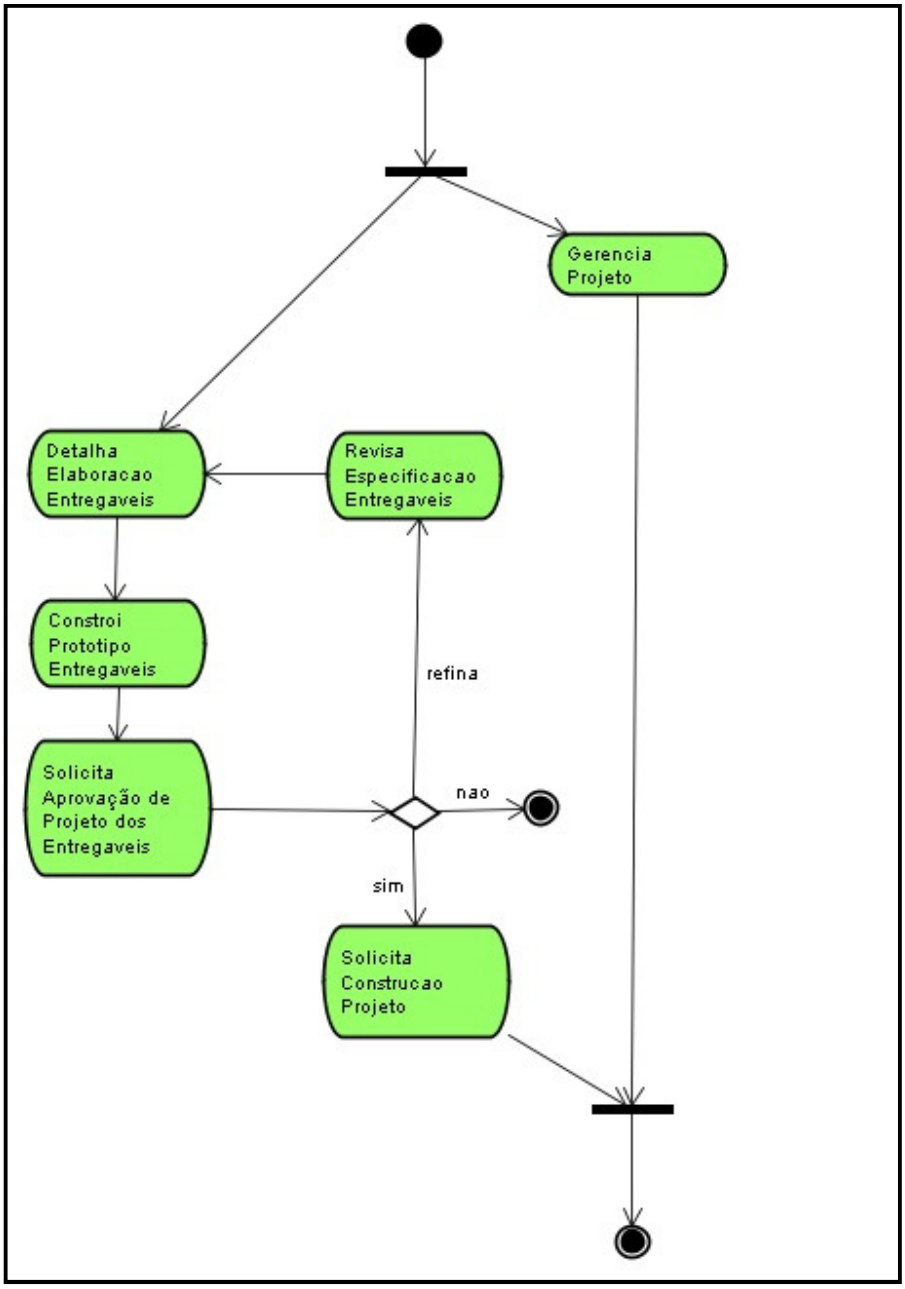
Processo de Análise de Negócio



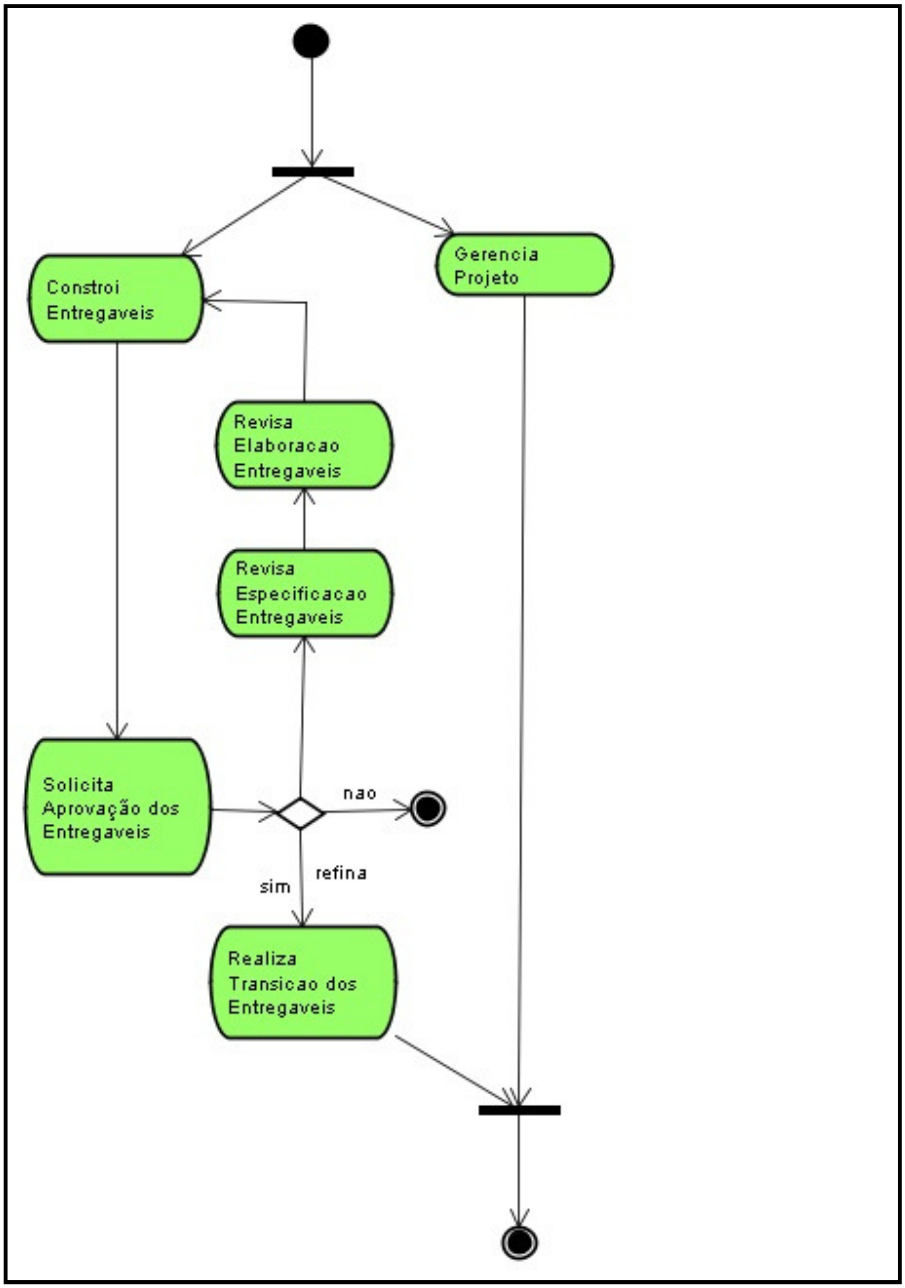
Processo de Concepção



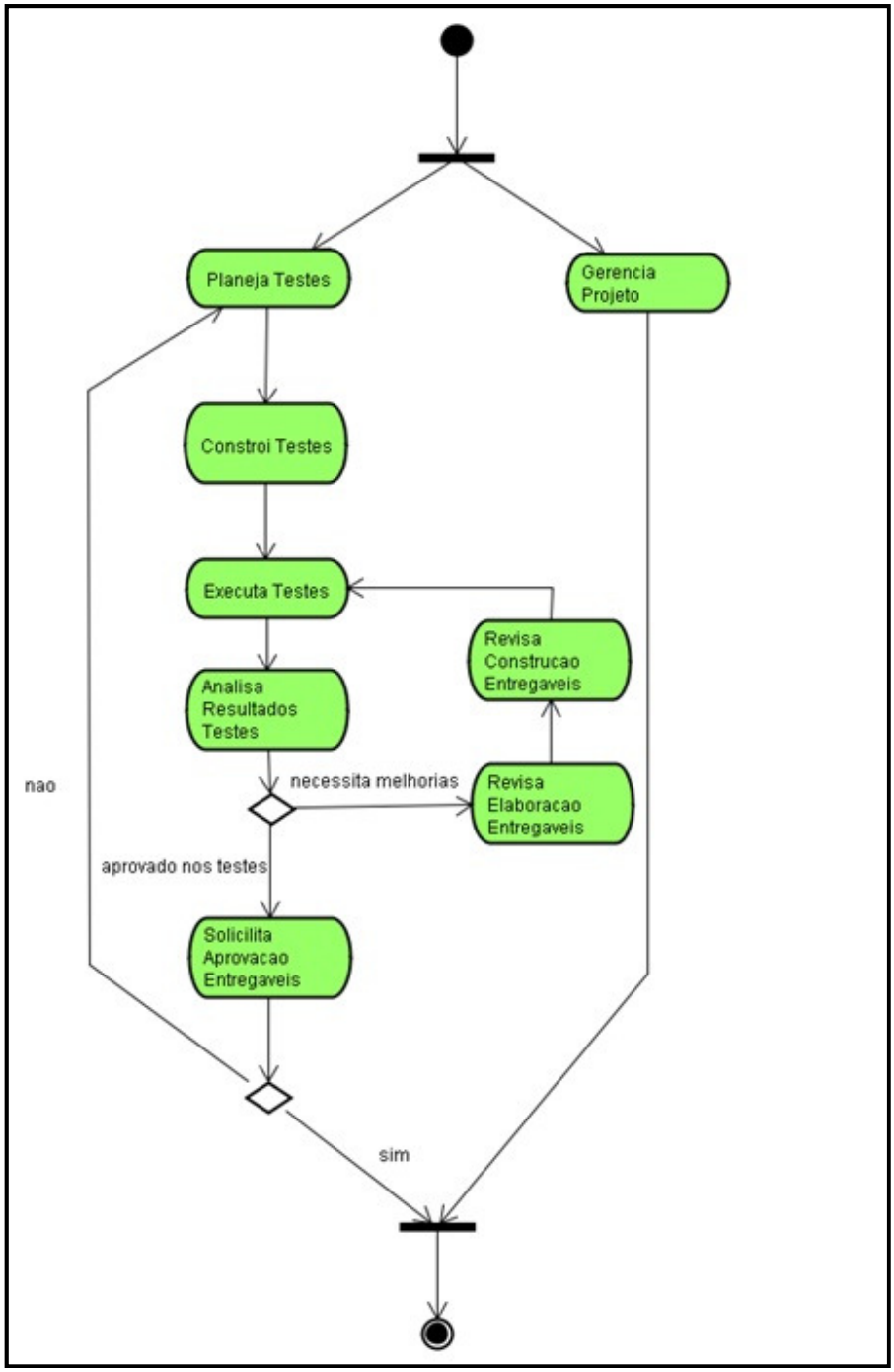
Processo de Planejamento



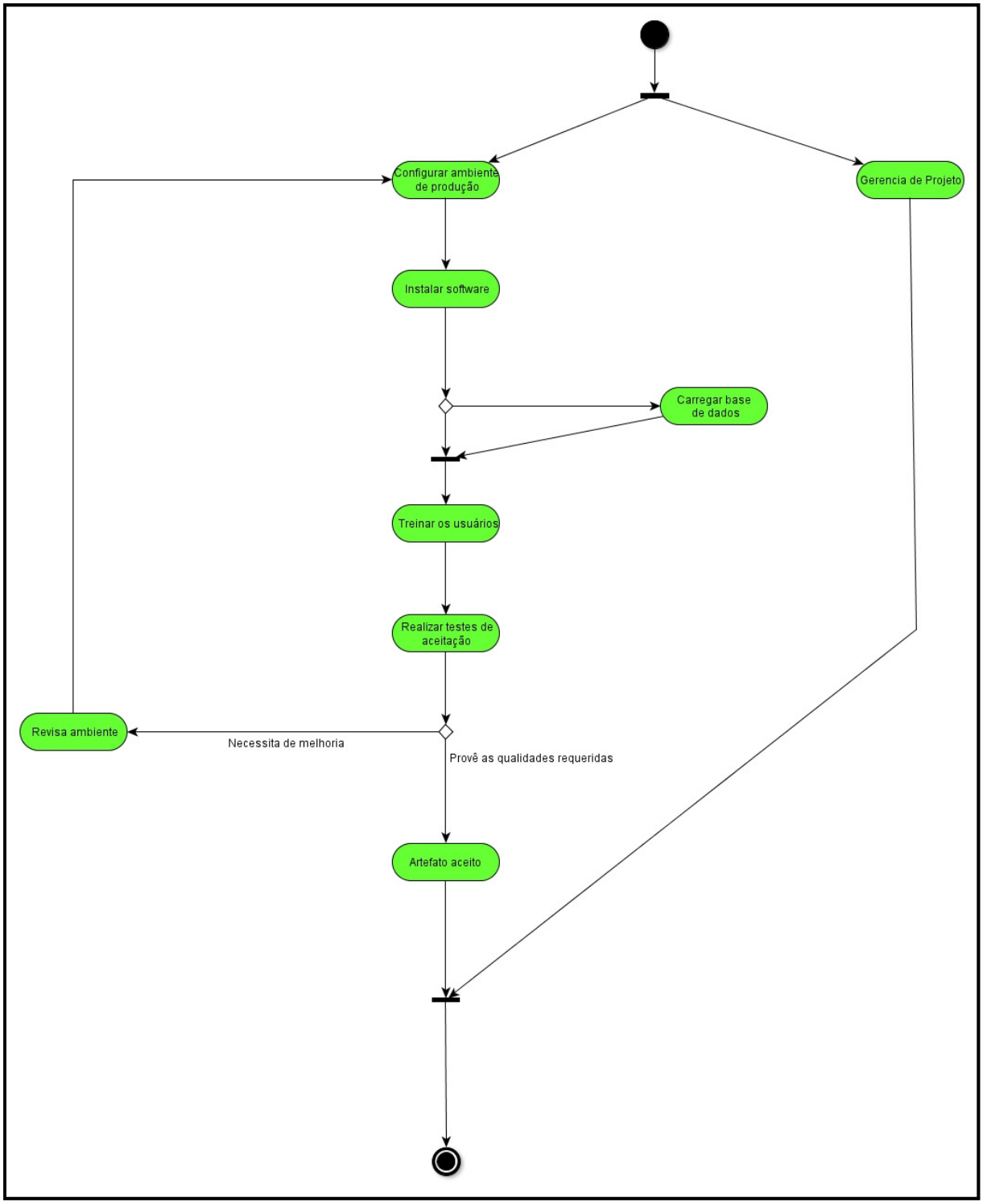
Processo de Construção



Processo de Testes

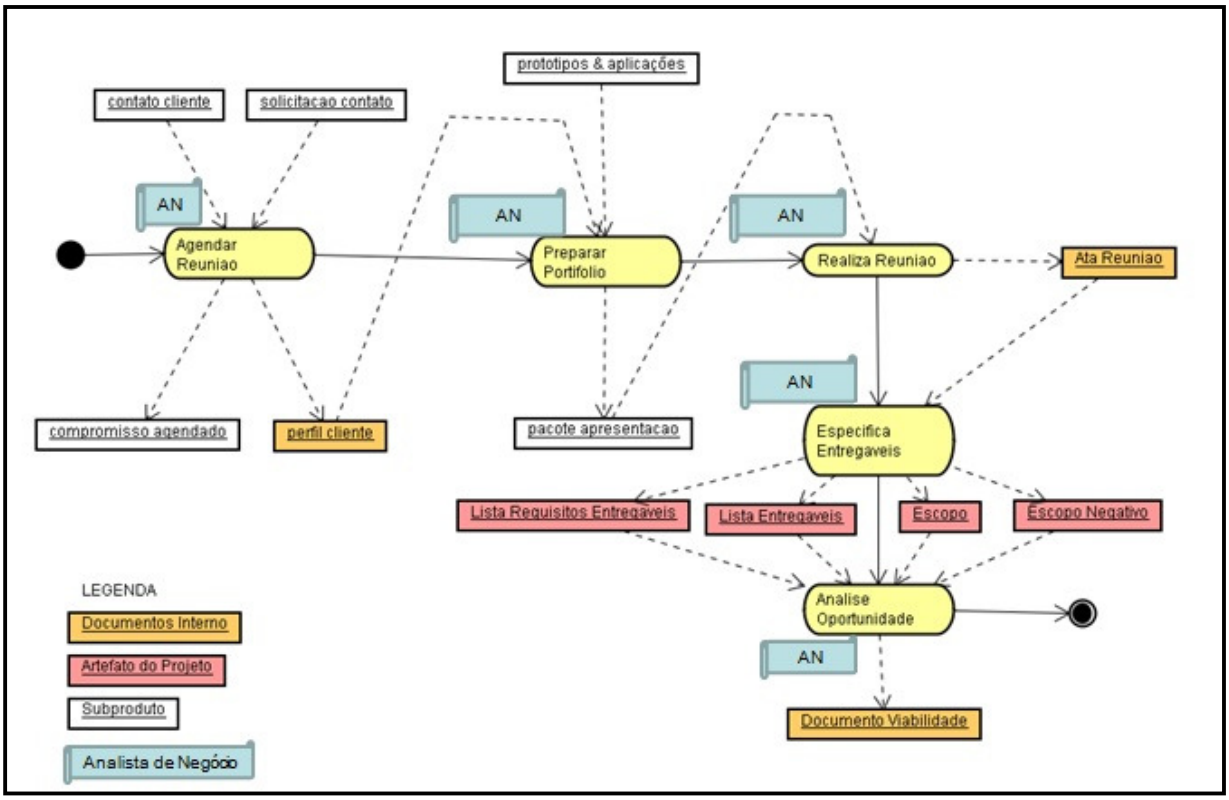


Processo de Implantação

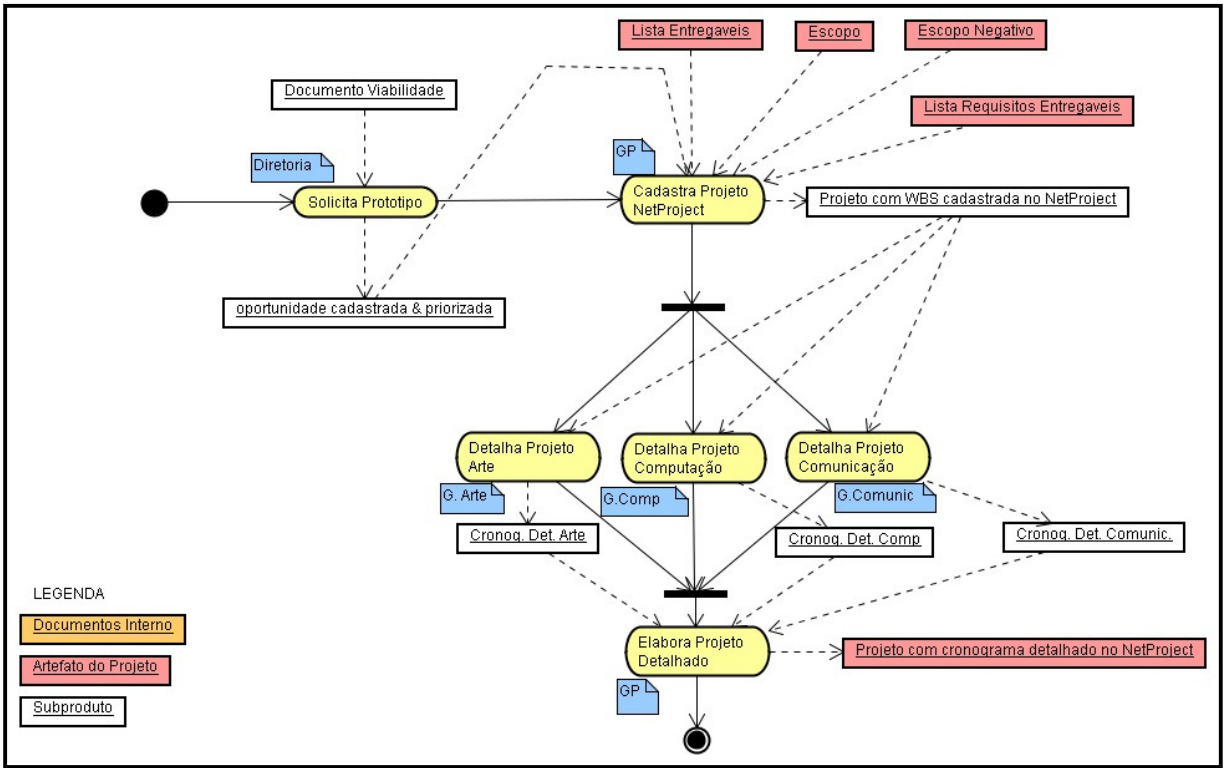


Processo de Análise de Negócio

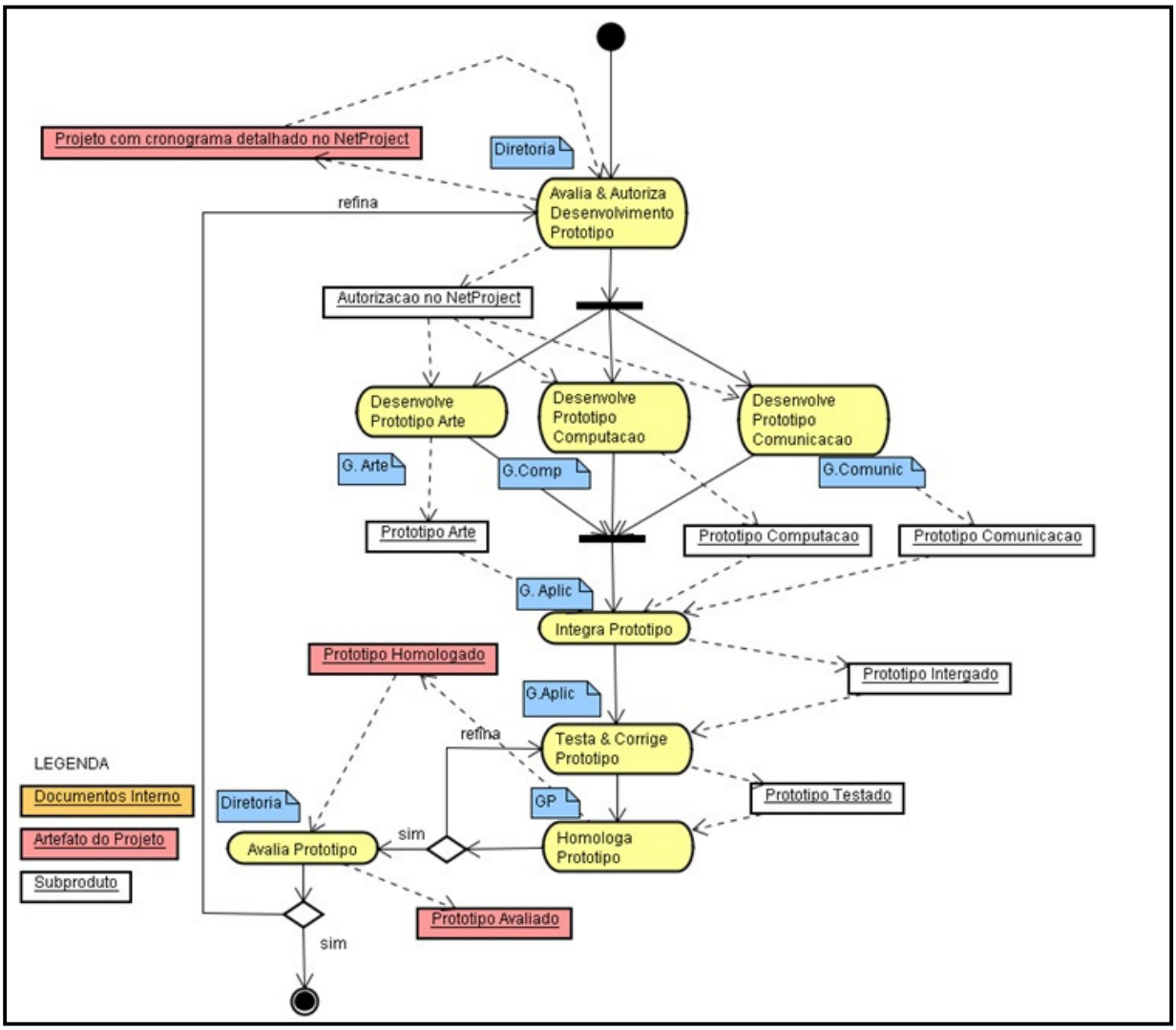
Atividade 1: Reunir com Cliente



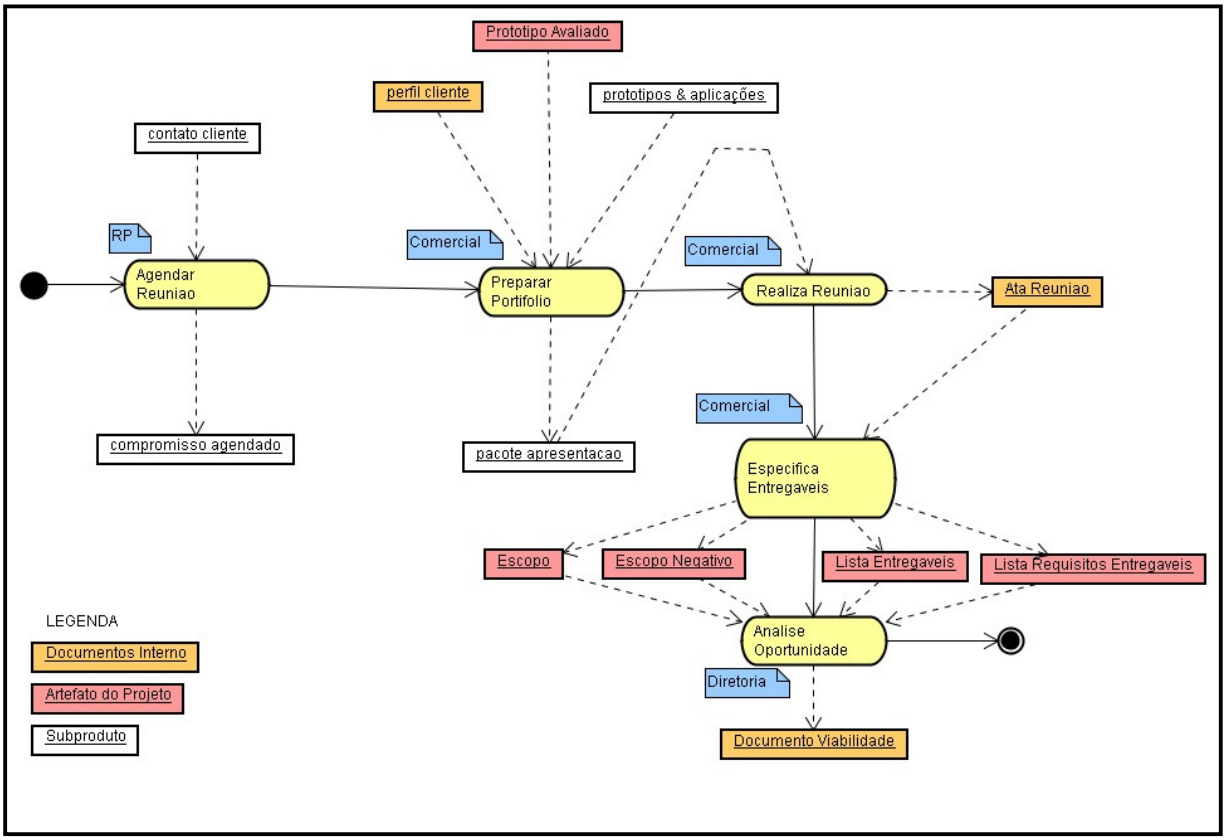
Atividade 2: Elabora Projeto Protótipo



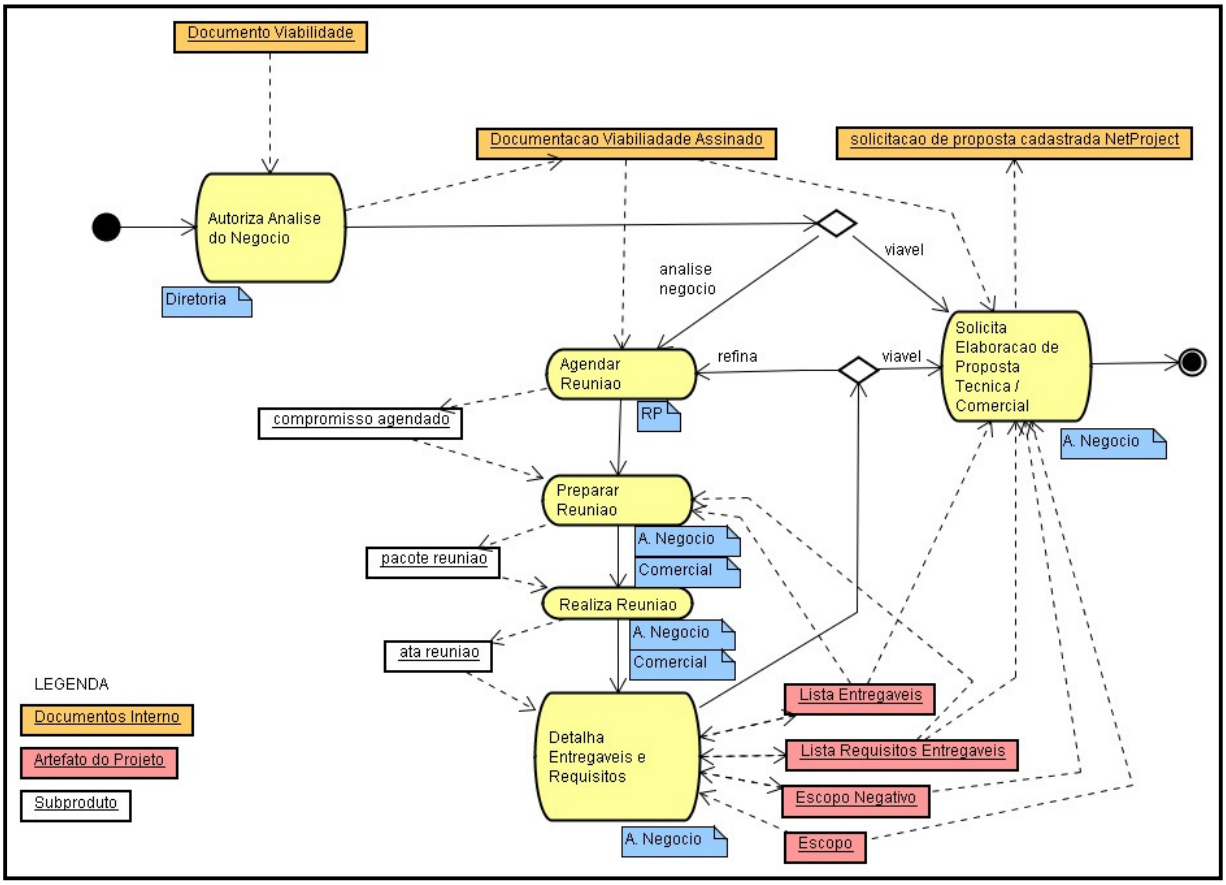
Atividade 3: Desenvolve Protótipo



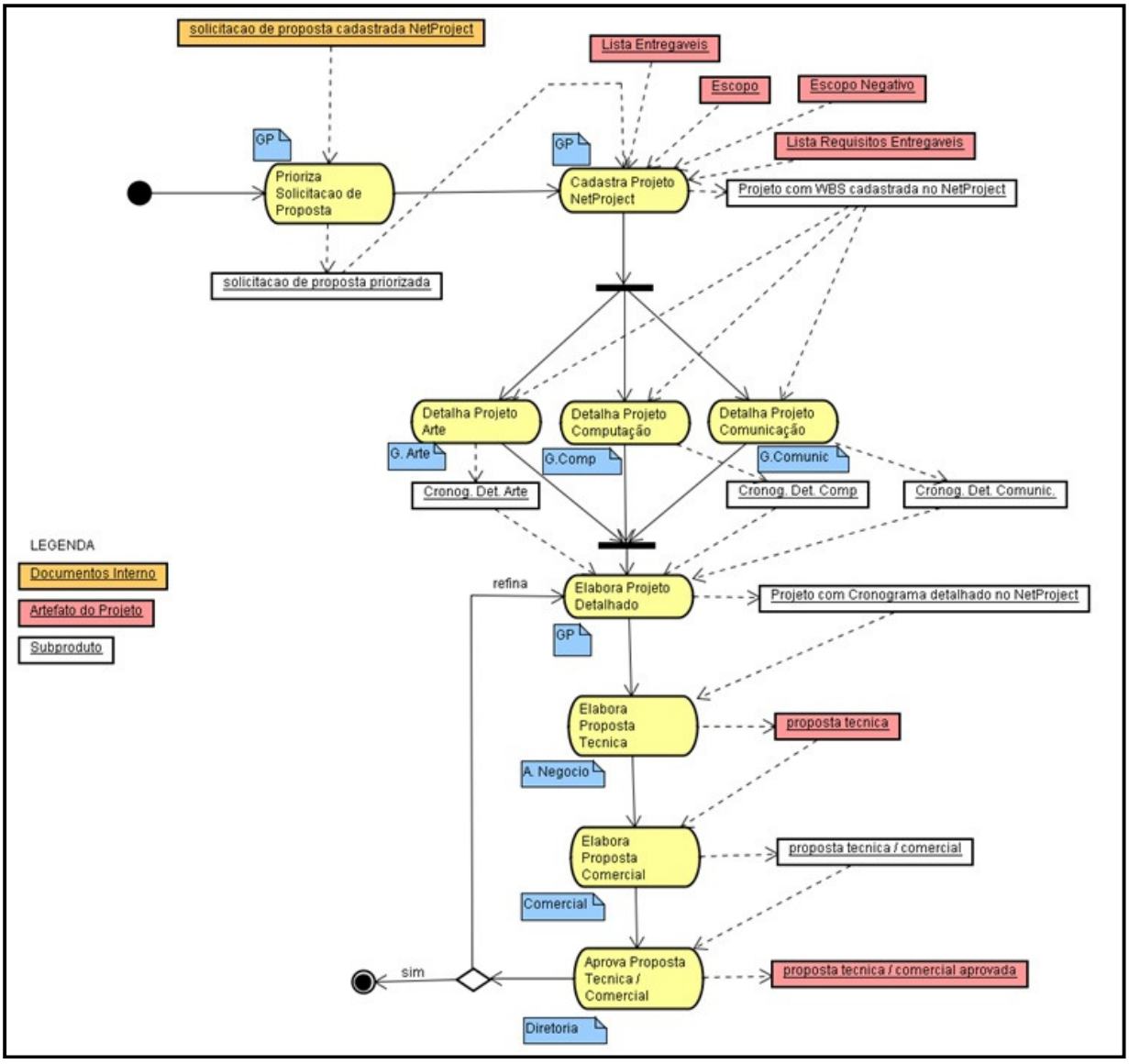
Atividade 4: Apresenta Protótipo para Cliente



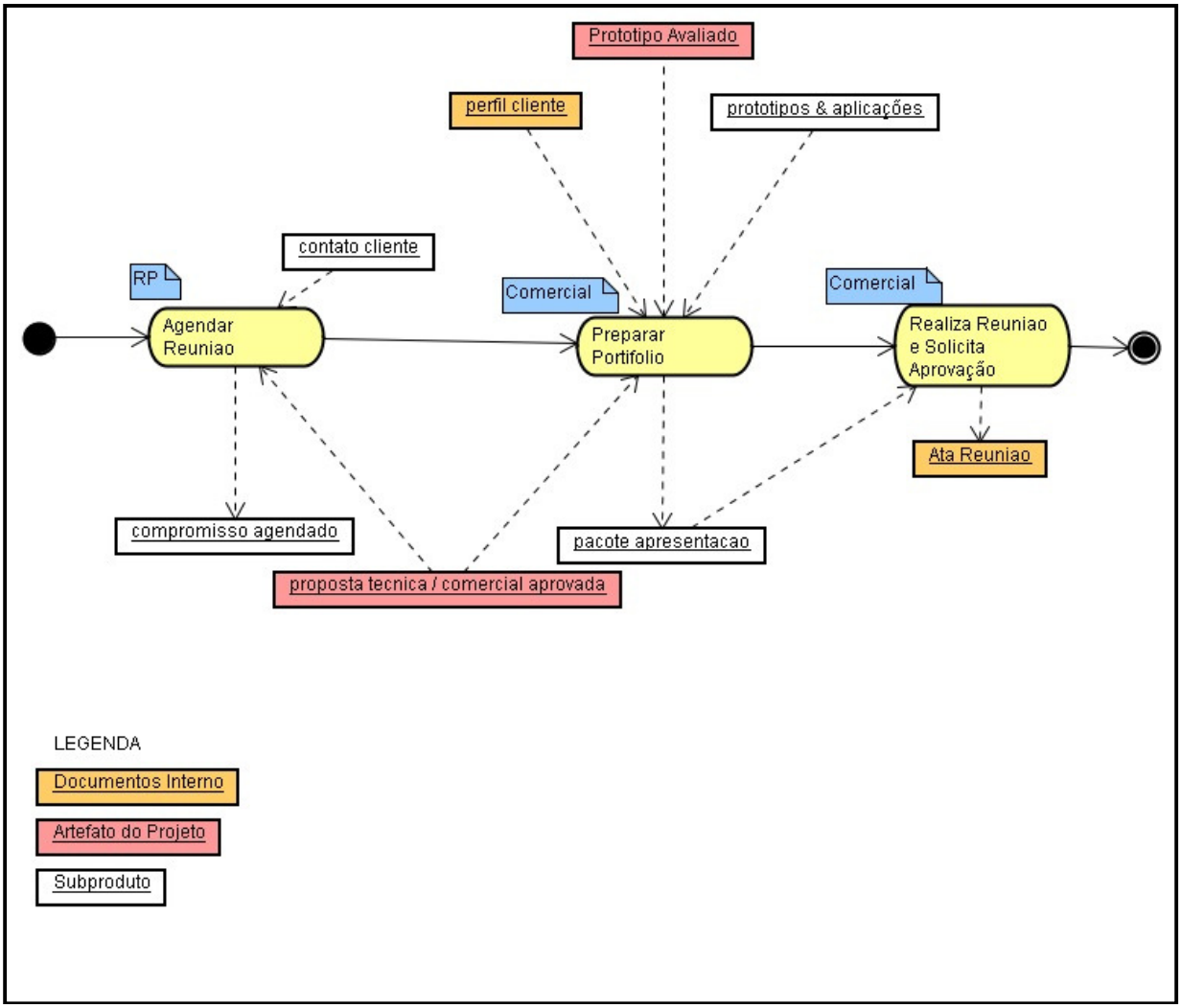
Atividade 5: Especifica Entregáveis



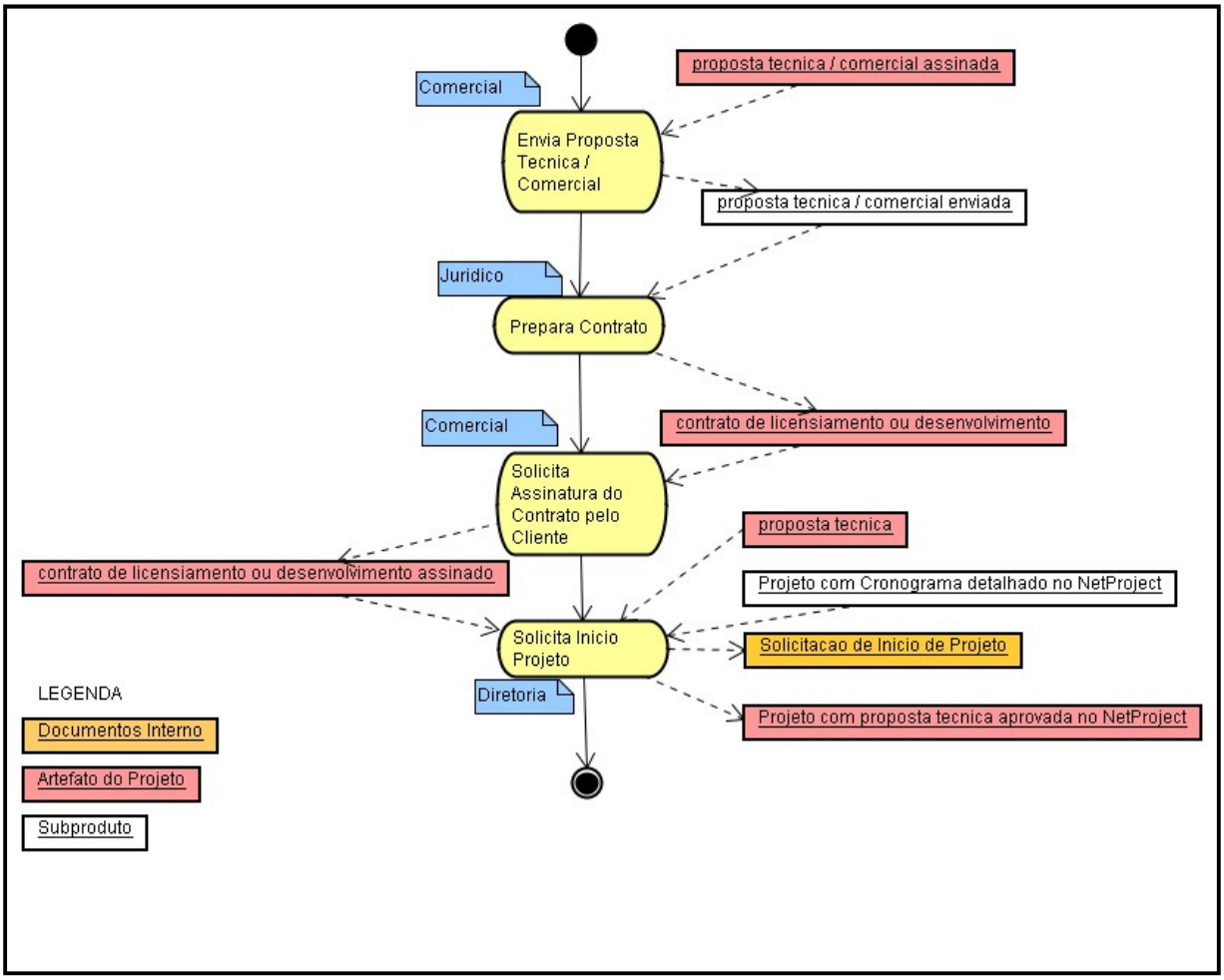
Atividade 6: Elabora Proposta Técnica/Comercial



Atividade 7: Solicita Aprovação de Proposta para o Cliente



Atividade 8: Solicita Início do Projeto



ANEXO II

ARTEFATOS DE SOFWTARE



<Nome do Projeto>

Especificação de Requisitos

Versão <1.0>

Histórico da Revisão

Data	Versão	Descrição	Autor
<dd/mmm/aa>	<x.x>	<detalhes>	<nome>

Índice

1. Objetivo 19
2. Descrição do Produto 19
   2.1 Escopo do Produto 19
      2.1.1 Nome do produto e de seus componentes principais 19
      2.1.2 Missão do produto 19
      2.1.3 Limites do produto 19
      2.1.4 Benefícios do produto 19
   2.2 Serviço oferecidos pelo produto 20
      2.2.1 Diagrama de contexto 20
      2.2.2 Descricão dos Servicos 20
      2.2.3 Generalizacão dos Atores 21
      2.2.4 Descricão dos Atores 21
3. Definições e Siglas 23
4. Requisitos de Interface 23
   4.1 Interfaces de Usuário 23
      4.1.1 Janela Principal 24
   4.2 Interfaces de Hardware 25
      4.2.1 Interface com o Servidor de Documentos 25
   4.3 Interfaces de Software 26
      4.3.1 Interface com Diretório de Mapas 27
5. Requisitos Funcionais 28
   5.1 Requisito 1 – Amazenar e recuperar imagens de satélite 28
   5.2 Requisito 2 – Visualizar imagens de satélite 28
6. Requisitos Não Funcionais 28
   6.1 Usabilidade 28
   6.2 Confiabilidade 29
   6.3 Desempenho 29
   6.4 Manutenibilidade 30
   6.5 Portabilidade 31
   6.6 Requisitos Legais 31
   6.7 Requisitos de Segurança 31
   6.8 Outros Requisistos Não Funcionais 31
7. Referências 32

Especificação de Requisitos

1 Objetivo

Descreve-se aqui o propósito desse documento.

Geralmente, o objetivo da especificação de requisitos é apresentar os requisitos contempláveis ou não contempláveis que são importantes para o desenvolvimento de um sistema. Os requisitos determinam o que um sistema deve fazer, em outras palavras, do que o sistema deve ser capaz ou quais são os servicos por ele oferecidos. Requisitos podem ser também definidos como os critérios de aceitacão do cliente com relacão ao sistema encomendado. Portanto, estes requisitos devem ser levantados pela equipe do projeto, em conjunto com representantes do cliente, usuários chaves e outros especialistas da área de aplicação.

Os requisitos também determinam como o software interage com as pessoas, com o hardware do sistema, com outros sistemas e com outros produtos?

Os requisitos podem ser do seguintes tipos:

- Requisitos de negócio
- Requisitos legais e normativos
- Atributos de qualidade do sistema: usabilidade, performance, suportabilidade, confiabilidade
- Outros requisitos de ambiente, compatibilidade e restrições do projeto

Requisitos de alta qualidade são claros, completos, sem ambigüidade, implementáveis, consistentes e testáveis. Os requisitos que não apresentem estas qualidades são problemáticos: eles devem ser revistos e renegociados com os clientes e usuários.

2 Descrição do Produto

a. Escopo do Produto

i. Nome do produto e de seus componentes principais

Nome do produto a ser produzido e seus componentes, se houver.

ii. Missão do produto

Descreve-se aqui os objetivos do produto que deverá ser desenvolvido.

De preferência, usar um único parágrafo que sintetize a missão a ser desempenhada pelo produto: ou seja, que valor o produto acrescenta para o cliente e os usuários.

iii. Limites do produto

Descreve-se aqui o que o produto não fará

O principal objetivo é evitar falsas expectativas por parte do cliente e usuários e explicitar requisitos adiados.

iv. Benefícios do produto

Identifica-se aqui os benefícios que o usuário terá com o produto.

Associar funções e benefícios torna possível fazer a priorização dos requisitos funcionais.

b. Serviço oferecidos pelo produto

i. Diagrama de contexto

Inclui-se aqui o diagrama de contexto do sistema.

Isto é, um diagrama de casos de uso que mostre as interfaces do sistema em seu ambiente de operacão. É importante apresentar os diversos atores (tipos de usuários) que iteragem com o sistema, sejam esses atores pessoas, hardware ou outro sistema. Cada caso de uso corresponde a um servico (macro-funcionalidade) oferecido pelo sistema.

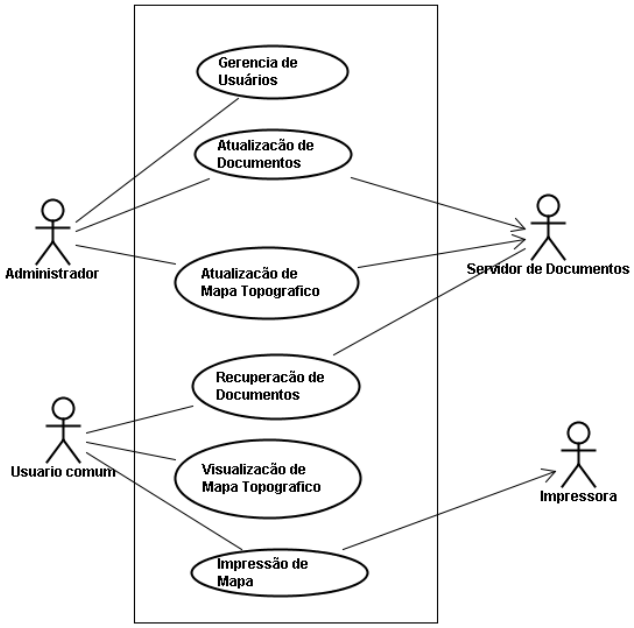


Figura 1 – Diagrama de Contexto do Sistema

.

ii. Descricão dos Servicos

Identifica-se aqui as principais funções que o produto desempenhará.

Descreve-se de forma consisa e objetiva cada serviço. Cada servicO corresponde a um dos casos de uso presentes no diagrama de contexto.

Tabela 1: Lista de Servicos

Número	Caso de Uso	Descrição
- 1	- Visualização de mapa topográfico	- O usuário carrega um mapa de topografico ele é apresentado na janela principal do sistema
- 2	- Atualizacão de documentos	- O usuário autentica-se no servidor do sistema e cria/atualiza arquivos no formato .doc

Número	Caso de Uso	Descrição
- 3	-	-

iii. Generalizacão dos Atores

Identifica-se aqui as principais relacões entre os atores que iteragem com o sistema

Descreve-se na forma de um Diagrama de Caso de Uso a relacões de herenca existentes entre os atores.

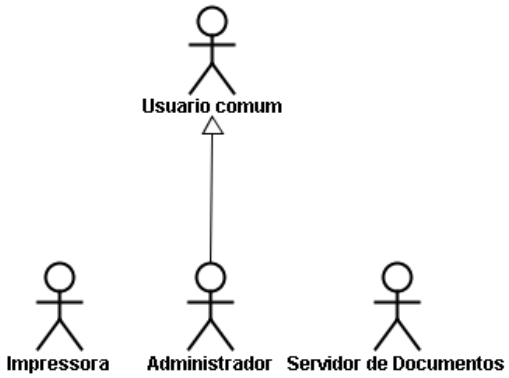


Figura 2 – Diagrama de Atores do Sistema

iv. Descricão dos Atores

Identifica-se aqui os principais atores que iteragem com o sistema

Descreve-se de forma consisa e objetiva cada ator. O objetivo é descrever os diferentes papéis que o usuário pode desempenhar quando interage com o sistema.

Tabela 2: Lista de Atores

Número	Ator	Descrição	Nivel de Instrução	Proficiência na Apliação
- 1	- Administrador	- Usuário com privilégios de escrita no sistema, pode criar/atualizar maps e documentos e é responsavel por gerenciar o cadatro de usuários do sistema	- Curso superior	- Alta
- 2	- Usuario comum	- Usuário sem previlégios de escrita, pode somente recuperar mapas e documentos no sistema	- Segundo grau completo	- Média

Número	Ator	Descrição	Nivel de Instrução	Proficiência na Apliação
- 3	-	-	-	-

3 Definições e Siglas

Descreve-se aqui a definição de todas as siglas, abreviações e termos usados no documento.

Deve-se supor que este documento será lido tanto por desenvolvedores, quanto por usuários. Por isso o documento deve conter as definições relevantes tanto para os termos do domínio de aplicação, quanto para termos técnicos utilizados e que não são de conhecimento do público geral.

Tabela 3: Definições e Siglas

Sigla ou Termo	Descrição
-	-
-	-
-	-
-	-

4 Requisitos de Interface

Descreve-se nas proximas secões de textos os requisitos relacionados à interface com o usuário e as interfaces com demais componentes externos, sejam estes de software ou hardware.

a. Interfaces de Usuário

Enumera-se em uma tabela as interfaces com o usuário e os atores que a elas terão acesso. Todas as intefaces devem ser brevemente descritas.

Tabela 4: Lista de Interfaces de Usuário

Número	Interface	Atores	Casos de Uso	Descrição
- 1	- Janela Principal	- Usuário comum.	- Visualizacão de Mapa Topografico.	- Janela principal do sistema, onde o menu principal e os mapas topograficos serão apresentados
- 2	- Janela de Cadastro de Usuários	- Administrador	- Gerência de Usuários	- Usuário sem previlégios de escrita, pode somente recuperar mapas e documentos no sistema
- 3	-	-	-	-

Descreve-se então os detalhes de layout das interfaces com o usuário do sistema.

Para cada interface com o usuário (tela, janela ou caixa de dialogo) é necessario descrever os campos nelas presentes e os comandos de acesso, por exemplo, clique com mouse, botões na barra de ferramentas e/ou teclas de atalhos.

i. Janela Principal

1. Layout

O software deverá executar a partir de uma janela do Windows apresentando na área central o mapa topografico do terreno em questão. Haverá uma barra de ferramentas contendo as funções onde o usuário poderá invocar.

2. Relacionamento com outras interfaces

Da interface principal será possível visualizar a interface de cadastro de equipamentos, a interface do painel de controle, a interface do inspetor de objetos e a interface de atualização de mapas topograficos.

3. Campos

Serão apresentados os seguintes campos:

- Barra de ferramentas:
  - Atualização de terreno – contém lista de mapas topográficos;
  - Barra de status – ativa/desativa a exibição da barra de status;
  - Painel de Controle – exibe/oculta a janela com a listagem de equipamentos;
  - Inspetor de Objetos – exibe/oculta a janela do inspetor de objetos;
- Área de desenho: região da tela onde é exibido o mapa topografico do terreno;
- Barra de Status: apresenta informações sobre a execução do programa e coordenadas de pontos indicados no terreno.

4. Comandos

A Tabela apresenta os comandos relacionados com a interface principal do software.

Tabela 5: Lista de Comandos – Janela Principal

Nome	Ação	Atalho
- Atualização de Terreno	- Lista os mapas topográficoes disponíveis para que seja escolhido aquele que ser exibido na área de desenho.	- Barra de ferramentas
- Exibição da Barra de Status	- Exibe ou esconde ou a barra de status.	- Barra de ferramentas
- Painel de Controle	- Exibe ou esconde o painel de controle.	- Barra de ferramentas e teclas [ctrl] + [P]

Nome	Ação	Atalho
- Inspetor de Objetos	- Apresenta o estado das variáveis monitoradas sobre o objeto selecionado.	- Barra de ferramentas

b. Interfaces de Hardware

Descreve-se aqui a comunicação do sistema com algum hardware, se houver.

Tabela 6: Lista de Interfaces de Hardware

Número	Interface	Atores	Caso de Uso	Descrição
- 1	- Interface de Leitura no Servidor	- Usuário comum.	- Visualizacão de Mapa Topografico. Recuperacão de Documentos	- Interface SAMBA com permissão apenas de leitura ao servidor de documentos para recuperacão de documentos e mapas topograficoes
- 2	- Interface de Escrita no Servidor	- Administrador	- Atualizacão de Documentos. Atualizacão de Mapa Topografico	- Interface SAMBA com permissão de escrita ao servidor de documentos, utilizada para armazenar documentos e mapas topográficos
- 3	- Interface de Impressão	- Usuário comum	- Impressão de Mapa	- Permite ao usuário enviar mapas para a impressora conectada ao sistema

Para cada interface de hardware é necessário descrever os atores que a utilizam, a fonte de dados entrada o destino dos dados de saída e o formato dos dados que nela trafegam, além dos relacionamentos com as demais interfaces do sistema.

i. Interface com o Servidor de Documentos

Será necessário uma conexão de rede, com privilégio de leitura, ao servidor que executa o diretório de documentos para aquisição de dados em diretório compartilhado.

1. Fonte de Entrada

Como o servidor documentos executa no sistema operacional Linux, os arquivos serão compartilhado via o software SAMBA.

2. Destino de Saída

Não se aplica.

3. Relacionamento com outras interfaces

Não se aplica.

4. Formato

Não se aplica.

c. Interfaces de Software

Descreve-se aqui a comunicação do sistema com algum outro sistema.

Tabela 7: Lista de Interfaces de Hardware

Número	Interface	Atores	Caso de Uso	Descrição
- 1	- Diretório de Mapas	- Administrador	- Atualizacão de Mapa Topografico	- Interface de integracão com o sistema "Diretorio de Mapas" da empresa XYZ através da qual mapas topograficos podem ser importados
- 2	-	-	-	-
- 3	-	-	-	-

Para cada interface de hardware é necessario descrever os atores que a utilizam, a fonte de dados entrada o destino dos dados de saída e o formato dos dados que nela trafegam, além dos relacionamentos com as demais interfaces do sistema

i. Interface com Diretório de Mapas

1. Fonte de Entrada

Os dados de entrada são fornecidos pelo sistema "Diretório de Mapas" versão 1.2 da empresa XYZ. Os dados deverão ser disponibilizadas por meio de um arquivo texto, de extensão .csv, disponibilizado em diretório compartilhado no Servidor de Documentos.

2. Destino de Saída

Os arquivos gerados serão copiados para o banco de dados do sistema após pré-procesamento para excluir dados replicados.

3. Relacionamento com outras interfaces

Os arquivos lidos serão utilizados para gerar os mapas topograficos na interface principal do sistema.

4. Formato

O arquivo .csv é um arquivo texto (ASCII), com valores separados por vírgula (formato csv – common separator value), no qual cada linha de texto é relativa ao isolinha do mapa topográfico (poligonal cotada), conforme o layout abaixo:

ID, x1, y1, z1, x2, y2, z2, x3, y3, z3, …

Onde:
ID → Identificador único da isolinha (poligonal cotada)
x1, y1, z1 → coordenadas tridimensionais para o primeiro vertice da isolinha (z1 = cota)
x2, y2, z2 → coordenadas tridimensionais para o segundo vertice da isolinha (z1 = cota)
x3, y3, z3 → coordenadas tridimensionais para o terceiro vertice da isolinha (z1 = cota)

Abaixo exemplifica-se o layout de uma linha desse arquivo para uma isolinha formada por 5 vertices na cota de 100 metros:

LIN4041,10,7, 100, 12, 8, 100, 19, 7, 100, 17, 15, 100, 9, 13, 100

O diretório deverá conter um único arquivo para cada mapa topografico. A seguinte nomenclatura deve ser utilizada para os arquivos em formato .csv: “topo_aammdd.csv”, onde “topo” é um prefixo utilizado em todos os arquivos que representam mapas topograficos, “dd” é dia no qual os dados foram coletados, “mm” é mês de coleta e “aa” é o ano. Exemplo: “gps260410t. csv” contém o mapa topografico levantado no dia 16 de abril de 2010.

5 Requisitos Funcionais

Descreve-se aqui as ações fundamentais através das quais o produto aceita e processa as entradas especificadas, gerando as respectivas saídas.

É feito o detalhamento desses requisitos, em nível suficiente para o projeto do produto, de seus testes de aceitação e de seu manual de usuário.

a. Requisito 1 – Amazenar e recuperar imagens de satélite

Descrição do requisito, prioridade relativa, nivel de dificuldade e critério de aceitacão.

O sistema deve ser capaz de armazenar e recuperar imagens de satelite em tabelas de bancos de dados relacionais.

Prioridade	(X) alta	( ) média	( ) baixa
Dificuldade	(X) alta	( ) média	( ) baixa

b. Requisito 2 – Visualizar imagens de satélite

Descrição do requisito, prioridade relativa, nivel de dificuldade e critério de aceitacão.

O sistema deve exibir as imagens de satélite armzenadas em banco em uma janela redimensionavel e permitir que o usuário navegue sobre ela. A imagem pode ser bem maior que a janela.

Prioridade	(X) alta	( ) média	( ) baixa
Dificuldade	( ) alta	(X) média	( ) baixa

6 Requisitos Não Funcionais

Descreve-se aqui os requisito não funcionais do sistema.

Os requisitos não funcionais, em geral, estabelecem o nivel de qualidade que os requisitos funcionais devem atender, ou condicionam e ambientam tais requisitos. Eles definem, por exemplo, critérios de usabilidade, portabilidade e desempenho que devem ser satisfeitos pelos requisitos funcionais.

a. Usabilidade

São os requisitos que definem as facilidades de uso do sistema, o nível de consistência dos dados apresentados e de documentação.

a. **Conformidade com padrões** – O sistema deve seguir o padrão XYZ – Qualidade de Serviços de Informação na empresa AB**C** e, quando necessário, o RUP – Rational Unified Process. Estes padrões podem ser adaptados.
b. **Nível de habilidade do usuário** – O sistema deve atender desde ... até os funcionários da empresa AB**C**, assim, o nível dos usuários que utilizarão o sistema deve ser considerado variado. Por este motivo, deve-se optar por interfaces com recursos gráficos e formulários de fácil compreensão, para que as tarefas possam ser realizadas no menor tempo e com o menor custo de treinamento.
c. **Presença de ferramentas de auxílio** – O sistema deve prover ao usuário helps on-line, menus, tool tips, enfim, documentação com o intuito de auxiliar em sua utilização.
d. **Treinamento** – Haverá necessidade de treinamento para usuários com variados níveis de habilidade, desde ... até analistas, supervisores e gerentes de risco da empresa AB**C**. Assim, haverá necessidade de treinamento para os variados níveis, que serão identificados na próxima fase.
e. **Dicas para o Usuário** – O sistema deve fornecer orientações a respeito do significado de cada campo apresentado na tela, visando auxiliar seu preenchimento ou utilização.

b. Confiabilidade

São os requisitos que definem a confiabilidade do sistema. Englobam aspectos como previsibilidade, acurácia de resultados, tolerância a falhas, recuperabilidade, dentre outros.

a. **Integridade dos dados** – O sistema deve manter a integridade dos dados residentes nas estações servidoras e também nas aplicações clientes para cadastros que são realizados off-line e depois transmitidos em lotes para as estações servidoras. Assim, é necessário manter, com integridade, as bases de dados centrais (estações servidoras) e locais (aplicações cliente).
b. **Controle de redundância** – O sistema deve possuir controle de redundância dos dados, uma vez que muitas informações estarão duplicadas nas estações servidoras e nas aplicações clientes. Visa-se desta forma, manter as bases de dados das estações servidoras e as bases de dados das aplicações clientes atualizadas e sincronizadas.
c. **Disponibilidade** – O sistema deve estar disponível 24 horas por dia, 7 dias por semana.
d. **Medidas de Tempo entre falhas** – Pode ocorrer uma falha do sistema a cada ano. O tempo total de falha esperado no ano é de 4 horas.
e. **Medidas de tempo de reparo** – Após uma falha, o sistema pode permanecer indisponível, no máximo, por um período de meia hora.
f. **Máximo de defeitos ou Taxa de defeitos** – A ser identificado na próxima fase.
g. **Rotinas Operacionais** – O backup dos dados sensíveis deve ser realizado diariamente. Será adotada a abordagem de backup incremental ou diferencial nos dias úteis e um backup total nos finais de semana. Os demais dados serão copiados de forma integral nos finais de semana. Dados sensíveis são os dados considerados imprescindíveis para operação do sistema, serão identificados na fase posterior, a fase de modelagem de dados.

c. Desempenho

Nesta seção são relatados os requisitos que melhor definem o desempenho do sistema. Engloba considerações sobre tempo de resposta, taxa de servico, velocidade de processamento, eficiência, precisão, consumo de recursos computacionais e volume de produção.

a. **Tempo de resposta para uma transação** - O sistema implementa serviços utilizando a tecnologia Web. Serviços implementados na Web devem ter um tempo de resposta

mínimo, não podendo ultrapassar os 10 segundos que já são praticados por corporações de grande porte que mantém serviços na Web.

b. **Número de usuários do sistema distribuído ao longo do tempo** – Atualmente a Empresa & possui cerca de 2 mil funcionários na área T e cerca de 30 mil parceiros cadastrados. Planeja-se um crescimento do número de parceiros em torno de 5% ao ano. Já o número de funcionários cresce a uma taxa menor, podendo variar entre 1% e 3% de crescimento anual.
c. **Estimativas de transações com o banco de dados** – Estima-se para o sistema que se tenha aproximadamente 800 mil operações de cadastro (consulta, alteração, inserção e exclusão) por ano. O crescimento destas operações não deve ultrapassar a 5% ao ano.
d. **Quantidade de acessos simultâneos** – O sistema deve suportar até 1 mil usuários simultâneos e uma taxa de 1% de crescimento anual.
e. **Quantidade de transações por segundo** – Supondo 21,5 milhões de operações com o banco anualmente. Extrapolando outras 21,5 milhões de operações que não acessam ao banco, tem-se cerca de 43/6,912 = 6,23 transações por segundo.
f. **Capacidade** – O sistema deve acomodar não simultaneamente os 32 mil usuários e simultaneamente os 7 mil usuários definidos anteriormente. Além disto, o sistema deve ter a capacidade de alocar recursos extras, proporcionais à taxa anual de crescimento de usuários.
g. **Área de armazenamento** – Supondo que uma operação de cadastro, seja esta de Produto ou Serviço, ocupe 1024 bytes (1KB). Já uma operação de @ ou $ ocupa, em média, 10K. A área de armazenamento mínima anual deve ser de 11 GBytes para cadastros e outros 110Gbytes para @´s e $´s. A área de espaço ocupada não leva em conta o fonte do sistema e outros dados necessários ao sistema como banco de regras, banco de técnicas de IC, banco de configurações de técnicas de IC, banco com os metadados do sistema, o Data Warehouse do sistema, entre outros. Este espaço será definido na segunda fase do projeto, após a modelagem de dados.

d. Manutenibilidade

São os requisitos que definem a capacidade do sistema de suportar mudanças, evoluções e reparos. Definem a testabilidade, extensibilidade, adaptabilidade, compatibilidade, entre outros.

a. **Padrões de programação** – Será adotado e adaptado o Java Programming Guideline, proposto por Scott Ambler.
b. **Padrões Gerais** – Os layouts do sistema, por exemplo, Web ou DESKTOP, deve seguir o mesmo padrão adotado para o sistema RTW da empresa XYZ, feito em Java.
   A arquitetura do sistema será baseada no modelo MVC (Model View Controller).
c. **Ferramentas de avaliação de impacto** – Deverá ser adotado mecanismos propostos pela disciplina de Gerência de Configuração e Mudanças do RUP.
d. **Características de extensibilidade de linguagem adotada** – Os componentes a serem desenvolvidos para o sistema devem possuir extensibilidade, ou seja, devem facilitar a adição de novas características que se fizerem necessárias.
e. **Facilidades de instalação** – O sistema deve possuir mecanismos de facilitação de instalação. Ressalta-se, ainda, que esta característica é fundamental, pois os usuários são pessoas com pouco conhecimento de informatica. Assim, por exemplo, os módulos desktop dos cadastros devem possuir uma facilidade de instalação similar a do sistema RTW da empresa XYZ.
f. **Elaboração e Distribuição de novas versões** – Deverá existir um gerenciamento para controlar a elaboração de novas versões do subsistema. Também deverá haver um controle para a distribuição das últimas versões para as partes interessadas. Poderá ser adotado mecanismos propostos pela disciplina de Gerência de Configuração e Mudanças do RUP.
g. **Interoperabilidade** – O sistema deve considerar dados de entrada e saída que são mantidos em sistemas externos, ou seja, sistemas que estão fora do domínio do sub-

sistema. Assim, deverá haver interoperabilidade entre o sistema e sistemas legados. As regras para este inter-relacionamento deverão ser estabelecidas na próxima fase.

h. **Mídia de Armazenamento** – A mídia de armazenamento para os dados persistentes e temporários do sub-sistema serão discos rígidos (HD). Já para os dados de backup, a mídia será definida na próxima fase.

e. Portabilidade

Descreve-se aqui restricões relacionadas a infra-estrutura tecnológica necessária para o desenvolvimento, implantacão e a utilização do sistema.

Por exemplo:

- Linguagem de programação – Java, HTML e JavaScript;
- Banco de dados – Oracle 10g;
- Servidor de Aplicação – JBoss;
- Sistema Operacional – RedHat 5.0 e
- Configuração de Hardware, Software Básico, Rede de Comunicação de Dados e Instalações Físicas para o desenvolvimento, as simulações e os testes do sistema na UFOP em conjunto com os representantes da empresa XYZ – A ser definido posteriormente.

Outros aspectos da infra-estrutura serão levantados na segunda fase do projeto.

f. Requisitos Legais

Descreve-se aqui os requisitos não funcionais derivados da legislação que regula a construção do sistema, que restringem ou controlam de alguma maneira o seu desenvolvimento.

Por ser a atividade fiscal plenamente vinculada, se faz necessário observar os dispositivos da legislação pertinentes a cada caso. No tocante ao projeto, os requisitos legais a serem atendidos em primeiro lugar são os dispositivos normativos editados pela LEI RRR que dispõem sobre segurança de dados, entre elas segurança física e lógica dos mesmos, requisitos quanto a cadastramento de usuários e sistemas, além do uso de extratores e acesso ao ambiente externo.

g. Requisitos de Segurança

Descreve-se aqui os requisitos que definem a política de segurança adotada para o sistema.

a. **Mecanismos ou sistemas de controle de acesso** – A princípio foi definido que o sistema vai utilizar os serviços do sistema NNN para controle acesso e autenticacão de usuários.
b. **Transações e perfis de acesso** – O sistema vai se valer do perfil, transação e âmbito do usuário para permitir o acesso a serviços internos. O perfil pode ser um leque de opções razoáveis, dependendo da política a ser adotada. As transações são as descritas e implementadas no sistema. Os âmbitos podem ser: individual, local, regional e nacional.
c. **Sigilo** – Sabe-se que o sistema e os dados manipulados por este são de sigilo absoluto, mas os critérios e procedimentos que garantam tal sigilo serão especificados em detalhes na próxima fase do projeto.

h. Outros Requisitos Não Funcionais

Descreve-e aqui os requisitos identificados para o sistema que não estão mapeados dentro da classificação adotada no corpo desse documento.

Por exemplo: É necessário prover mecanismos de captura de dados e realização de cálculos estatísticos referentes aos seguintes itens:

- Apuração do tempo médio de resposta de módulos residentes nas estações servidoras.
- Consumo de memória das estações servidoras por parte do sistema.
- Quantidade de usuários simultâneos para o sistema instalados nas estações servidoras da empresa XYZ.

a. **Condições de Entrega** – As disposições deste item estão previstas nas cláusulas do convênio entre a UFOP e a QQQ.
b. **Orçamentária e Financeira** – Antes da assinatura do convênio e antes de ter sido empenhada pela administração, existe uma limitação de ordem orçamentária e financeira que é o contingenciamento dos recursos que poderiam vir a ser alocados para o desenvolvimento.
c. **Financeira** – Depois da assinatura do convênio, uma limitação financeira ao projeto é o instituto do empenho (instrumento no qual a administração pública se compromete a fazer o pagamento por um determinado compromisso).
d. **Econômicos** – Todas as compras devem ser feitas seguindo os trâmites da Lei nº 8.666 de 1993 e legislações pertinentes posteriores.
e. **Distribuição** – O sistema deverá possuir um servidor de aplicações. A arquitetura do sistema deverá ser dividida em camadas, base de dados centralizada.
f. **Monitoramento** – O sistema deverá controlar e gerenciar transações, transmissões e execuções realizadas em tempo de execução. Desta forma, deve manter logs pertinentes a cada um destes itens, para serem utilizados em auditorias, com o intuito de gerenciar e controlar o uso do sistema. Também deve ser possível identificar situações críticas durante a execução do sistema e tomar as ações necessárias para concluir corretamente ou investigar tais situações.
g. **Replicação de Bases de Dados** – O sistema deve prever um mecanismo de segurança que controle o espelhamento de Bases de Dados das estações servidoras. Deve-se possuir desta forma, um mecanismo que forneça tolerância a falhas, ou seja, no caso de uma base de dados falhar outra base de dados assuma o processamento de forma segura e transparente para o usuário.

7 Referências

Descreve-se aqui a informação necessária para que todas as fontes de dados citadas na ERSw possam ser recuperadas, caso necessário

A lista de referências deve ser completa e estar em acordo com as normas bibliográficas vigentes. Para cada referências é importante indicar, no mínimo, os autores, o título, o nome do veículo onde a referência foi divulgada e o ano da publicação.

ANEXO II



<Nome do Projeto>

Especificação de Casos de Uso

Versão <1.0>

Histórico da Revisão

Data	Versão	Descrição	Autor
<dd/mmm/aa>	<x.x>	<detalhes>	<nome>

Índice

1. Objetivo 36
2. Descricão do Produto 36
3. Definições e Siglas 36
4. Diagramas de Caso de Uso 37
   4.1 Diagrama de Contexto 37
   4.2 Nome do primeiro Diagrama de Caso de Uso 38
5. Detalhamento dos Casos de Uso mais Significativos 38
   5.1 UC000 - Nome do primeiro Caso de Uso 39
      5.1.1 Fluxo Básico de Eventos 39
      5.1.2 Fluxos Alternativos 39
      < A1 Primeiro Fluxo Alternativo > 39
      < A2 Segundo Fluxo Alternativo > 39
      5.1.3 Pré-Condições 39
      5.1.4 Pós-Condições 39
      5.1.5 Pontos de Inclusão 40
      5.1.6 Pontos de Extensão 40
      5.1.7 Requisitos Especiais 40
      < Primeiro Requisito Especial > 40
   5.2 UC001 - Edicão de Mapas Topograficos 40
      5.2.1 Fluxo Básico de Eventos 40
      5.2.2 Fluxos Alternativos 41
      < A1 Usuário Cancela Edicão > 41
      < A2 Usuário Altera Ferramenta > 41
      5.2.3 Pré-Condições 41
      5.2.4 Condições Posteriores 41
      5.2.5 Pontos de Inclusão 41
      5.2.6 Pontos de Extensão 41
      5.2.7 Requisitos Especiais 41
   5.3 UC002 - Selecionar Mapa 41
      5.3.1 Fluxo Básico de Eventos 41
      5.3.2 Fluxos Alternativos 41
      5.3.3 Pré-Condições 41
      5.3.4 Condições Posteriores 41
      5.3.5 Pontos de Inclusão 42
      5.3.6 Pontos de Extensão 42
      5.3.7 Requisitos Especiais 42
   5.4 UC003 – Transladar Mapa 42
      5.4.1 Fluxo Básico de Eventos 42
      5.4.2 Fluxos Alternativos 42
      5.4.3 Pré-Condições 42
      5.4.4 Condições Posteriores 42
      5.4.5 Pontos de Inclusão 42
      5.4.6 Pontos de Extensão 42
      5.4.7 Requisitos Especiais 42
6. Referências 42

Especificação de Casos de Uso

1 Objetivo

Descreve-se aqui o propósito desse documento.

Basicamente, o documento de especificação de casos de uso tem como objetivo descrever a relação de casos de uso do sistema que será projetado. Sua principal funcão é detalhar os requisitos funcionais de um sistema. A especificação de casos de uso descreve um sistema do ponto de vista dos usuários ou clientes, seja esse usuário uma pessoa, um hardware ou um outro sistema. Como um mesmo usuário pode desempenhar varios papeis ao interagir com o sistema, por exemplo, um usuário pode autenticar-se como o administrador do sistema ou como um usuário sem previlégios, casos de uso são descrito com relacão a esses papeis e não aos usuários.

Cada caso de uso representa uma interacão discreta entre um usuário e o sistema para a obtecão de um único servico por ele fornecido. Portanto, para descrever um casos de uso procura-se determinar e explicar a sequencia de interacões realizadas entre o usuário e o sistema afim de realizar uma tarefa. Desta forma, um casos de uso é descrito por um fluxo principal de interacão que, em geral, descreve a maneira mais facil e direta de se obter um servico e, muitas vezes, por fluxos alternativos que, frequentemente, procuram descrever a forma na qual excecões serão tratadas.

Tanto o “Diagrama de Contexto” como os “Requisitos de Interface” decritos no documento “Especificacão de Resquisitos” são excelentes fontes de informacão sobre os casos de uso de um sistema.

2 Descricão do Produto

Faz-se aqui uma breve descrição sobre o produto, o contexto no qual ele se insere, e sobre o histórico do projeto.

O objetivo dessa secão de texto é tonar mais facil o entendimento do documento e situar o leitor.

3 Definições e Siglas

Descreve-se aqui a definição de todas as siglas, abreviações e termos usados no documento.

Deve-se supor que este documento será lido tanto por desenvolvedores, quanto por usuários. Por isso o documento deve conter as definições relevantes tanto para os termos do domínio de aplicação, quanto para termos técnicos utilizados e que não são de conhecimento do público geral.

Tabela 3: Definições e Siglas

Sigla ou Termo	Descrição

Sigla ou Termo	Descrição
-	-
-	-
-	-
-	-

4 Diagramas de Caso de Uso

Detalha-se aqui os relacionamento existentes entre os caso de uso e os atores do sistema.

Cada caso de uso identificado no “Diagrama de Contexto” contido no documento “Especificacão de Requisitos” pode ter dado origem a um ou mais requisitos funcionais. Todos os requisitos funionais que merecem ser detalhados devem ter o fluxo básico e os fluxos alternativos de seus casos de uso especificados. No entanto, enquanto os fluxos descrevem os detalhes de cada caso de uso, os “Diagramas de Casos de Uso” descrevem os relacionamentos dos casos de uso entre si e com os atores que interagem com o sistema. Interacões entre os atores não devem ser documentadas nos diagramas.

São duas as possiveis relacões entre os casos de uso: inclusão ou extensão. A relação de extensão indica que o comportamento do caso de uso estendido pode ser ou não inserido no caso de uso extensor. A notação é uma seta pontilhada partindo da extensão para o caso de uso estendido com a etiqueta <<extend>>. A relacão de inclusão implicanda que o comportamento do caso de uso incluído é obrigatoriamente inserido no comportamento do caso de uso inclusor. A notação é uma seta pontilhada que aponta para o caso de uso incluído com o estereótipo <<include>>..

Recomenda-se a documentacão de no mínimo o diagrama de contexto. Além disso, recomenda-se o uso de diagramas para detalhar partições do diagrama de contexto, mostrando grupos correlatos de casos de uso primários e os atores; diagramas de que mostrem todos os casos de uso de um determinado ator; diagramas para mostrar um certo caso de uso e seus relacionamentos; ou diagrama de uma versão do sistema ou prototipo.

a. Diagrama de Contexto

Isto é, um diagrama de casos de uso que mostre as interfaces do sistema em seu ambiente de operacão. É importante apresentar os diversos atores (tipos de usuários) que iteragem com o sistema, sejam esses atores pessoas, hardware ou outro sistema. Cada caso de uso corresponde a um servico (macro-funcionalidade) oferecido pelo sistema.

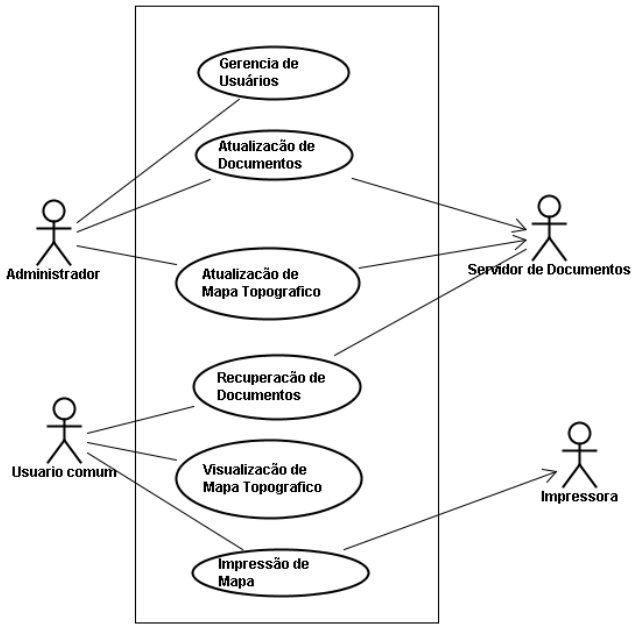


Figura 1 – Diagrama de Contexto do Sistema

b. Nome do primeiro Diagrama de Caso de Uso

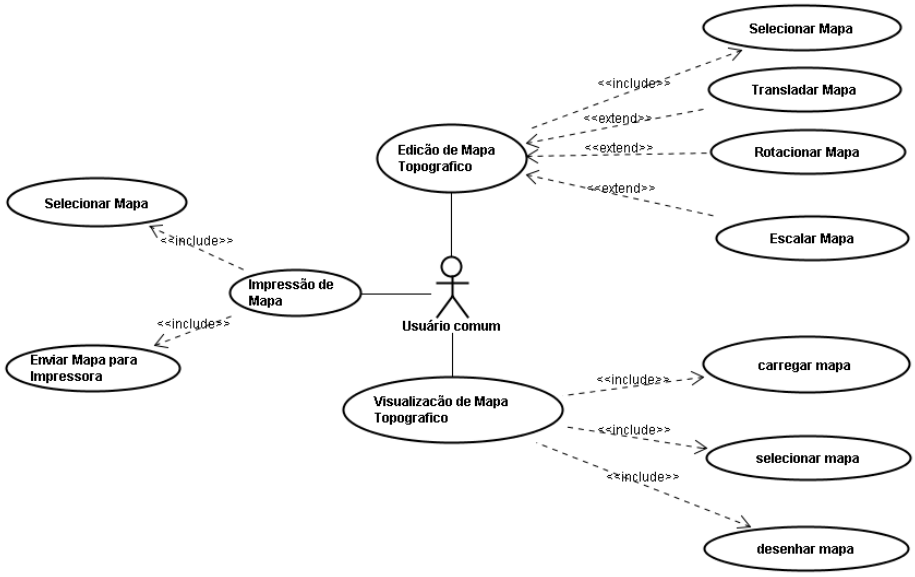


Figura 2 – Diagrama de Caso de Uso: Manipulacão de Mapas Topograficos por Usuário Comuns

5 Detalhamento dos Casos de Uso mais Significativos

Detalha-se aqui os casos de uso mais importantes entre aqueles presentes nos “Diagramas de Caso de Uso”. Para isso, descreve-se o fluxo básico e os fluxos alternativo desses casos de uso.

a. UC000 - Nome do primeiro Caso de Uso

i. Fluxo Básico de Eventos

O caso de uso desreve interacões entre um usuario e o sistema. Se forem trocadas informações, seja específico sobre o que é transmitido de um lado para outro. Por exemplo, não é muito esclarecedor dizer que o usuario digita informações do cliente se elas não forem definidas. É melhor dizer que o usuario digita o nome e o endereço do cliente.

Fluxos alternativos simples podem ser apresentados no texto do fluxo de básico. Se forem usadas apenas algumas sentenças para descrever o que acontece quando há uma alternativa, faça isso diretamente dentro do fluxo. Se o fluxo alternativo for mais complexo, utilize uma seção separada para descrevê-lo.
O fluxo complexo de eventos deve ser melhor estruturado em sub-fluxos. Ao fazer isso, a meta principal deve ser aprimorar a clareza do texto. Os subfluxos podem ser chamados muitas vezes de muitos lugares. Lembre-se de que o caso de uso pode executar subfluxos em seqüências opcionais, em loops ou mesmo vários ao mesmo tempo.

Uma imagem, às vezes, vale mais que mil palavras, entretanto, não há substituto para a prosa limpa e clara. Se aprimorar a clareza, sinta-se à vontade para colar fluxogramas, diagramas de atividades ou outras figuras no caso de uso. Se um fluxograma for útil para apresentar um processo de decisão complexo, use-o sem dúvida! O mesmo acontece para o comportamento dependente de estado, um diagrama de transição de estado freqüentemente explica o comportamento de um sistema melhor que página e mais páginas de texto. Utilize o meio de apresentação certo para o problema, mas tenha cuidado ao utilizar terminologia, notações ou figuras que o público-alvo pode não entender. Lembre-se de que seu objetivo é explicar, não confundir.

ii. Fluxos Alternativos

Comportamentos alternativos do sistemas são descritos em uma subseção do Fluxo Básico. Os fluxos alternativos geralmente descrevem como as exceções que ocorrem no fluxo principal serão tratadas pelo sistema.

< A1 Primeiro Fluxo Alternativo >

Descreva o fluxo alternativo, exatamente como qualquer outro fluxo de eventos.

< A2 Segundo Fluxo Alternativo >

Pode haver, e muito provavelmente haverá, vários fluxos alternativos em cada área de funcionalidade. Mantenha cada fluxo alternativo separado para aprimorar a clareza do documento.

iii. Pré-Condições

Insere-se aqui as condições prévias do caso de uso.

Uma condição prévia de um caso de uso descreve o estado em que sistema deve estar antes de um caso de uso ser executado. Pode haver várias condições prévias. Mantenha cada pré-condição separada para tornar o documento claro.

iv. Pós-Condições

Insere-se aqui as condições posteriores do caso de uso.

Uma condição posterior de um caso de uso descreve a lista de estados possíveis que o sistema pode estar imediatamente após um caso de uso ter sido concluído. Pode haver várias condições posteriores. Mantenha cada pós-condição separada para tornar o documento claro.

v. Pontos de Inclusão

Insere-se aqui os pontos de inclusão do caso de uso.

Definição local do ponto de inclusão do caso de uso.

vi. Pontos de Extensão

Insere-se aqui os pontos de extensão do caso de uso.

Definição local do ponto de extensão do caso de uso.

vii. Requisitos Especiais

Um requisito especial é, geralmente, um requisito não funcional que é específico de um caso de uso, mas não é fácil ou naturalmente especificado no texto do fluxo de eventos do caso de uso. Exemplos de requisitos especiais incluem requisitos legais e reguladores, padrões de aplicativos e atributos de qualidade do sistema a ser construído incluindo requisitos de utilidade, confiabilidade, desempenho ou suportabilidade. Adicionalmente, outros requisitos—como sistemas e ambientes operacionais, requisitos de compatibilidade e restrições de projetos—devem ser capturados nesta seção.

< Primeiro Requisito Especial >

Pode haver várias requisitos especiais. Mantenha cada requisito separado para tornar o documento claro.

b. UC001 - Edição de Mapas Topográficos

i. Fluxo Básico de Eventos

1. O usuário seleciona o mapa topográfico a ser editado em uma lista.
2. O usuário seleciona a ferramenta de edição: transladar, rotacionar ou escalar
3. O cursor do mouse é alterado de forma a refletir graficamente a transformação geométrica selecionada.
4. O usuário selecionar o ponto que será utilizado como origem cartesiana para a transformação geométrica que será aplicada, isto é, o ponto dentro do mapa que será considerado o ponto de (0,0).
5. O usuário movimenta o mouse sobre o mapa.
6. O sistema calcula os parâmetros das transformações geométricas de acordo com o vetor formado entre os dois pontos: translação – calcula a distância e direção, escala – calcula a distância e rotação - calcula o ângulo formado entre o vetor e o eixo horizontal.
7. O sistema representa graficamente o parâmetro sobre o mapa para que o usuário possa observá-lo.
8. O usuário seleciona o ponto que define os parâmetros utilizados nas transformacões geométricas.
9. A transformação geométrica é aplicada sobre o mapa topográfico
10. O mapa topográfico transformado é apresentado para o usuário.
11. O cursor do mouse é alterado de forma a refletir o estado ocioso do sistema.
12. O sistema retorna ao estado ocioso

ii. Fluxos Alternativos

< A1 Usuário Cancela Edição >

1. Após qualquer evento entre 2 e 7, o usuário pressiona a tecla “cancela”
2. O mouse passa a refletir graficamente o estado ocioso do sistema.
3. O sistema retorna ao estado ocioso

< A2 Usuário Altera Ferramenta >

1. Após o passo 2, o usuário seleciona outra ferramenta de edicão.
2. O sistema retorna ao fluxo básico a partir do evento 3.

iii. Pré-Condições

1. O mapa topográfico a ser editado deve ter sido aberto e carregado no sistema.
2. O sistema está ocioso.

iv. Condições Posteriores

1. O mapa topográfico transformado deve estar desenhado na janela principal do sistema.
2. O mouse passa a refletir graficamente o estado ocioso do sistema.
3. O sistema está no estado ocioso

v. Pontos de Inclusão

UC002 - Selecionar Mapa

vi. Pontos de Extensão

UC003 - Transladar Mapa
UC004 - Rotacionar Mapa
UC005 - Escalar Mapa

vii. Requisitos Especiais

Não se aplica.

c. UC002 - Selecionar Mapa

i. Fluxo Básico de Eventos

1. Usuário clica sobre o nome ou ícone do mapa topográfico em uma lista de mapas.
2. O sistema indica graficamente que o mapa foi selecionado.
3. O sistema desenha o mapa selecionado.

ii. Fluxos Alternativos

Não se aplica.

iii. Pré-Condições

1. O mapa topográfico a ser editado deve ter sido aberto e carregado no sistema.
2. O lista de mapa deve estar visível ao usuário.
3. O sistema está ocioso.

iv. Condições Posteriores

1. O mapa topográfico selecionado deve estar desenhado na janela principal do sistema.

2. O mouse passa a refletir graficamente o estado ocioso do sistema.
3. O sistema está no estado ocioso

v. Pontos de Inclusão

Não se aplica.

vi. Pontos de Extensão

Não se aplica.

vii. Requisitos Especiais

Não se aplica.

d. UC003 – Transladar Mapa

i. Fluxo Básico de Eventos

1. No passo 2 do fluxo de básico do caso de uso “UC001 - Edição de Mapas Topográficos” o usuário seleciona a ferramenta “Transladar”
2. No passo 6 do fluxo de básico do caso de uso “UC001 - Edição de Mapas Topográficos” o sistema calcula somente a distância e direção de translação.
3. No passo 9 do fluxo de básico do caso de uso “UC001 - Edição de Mapas Topográficos” o sistema calcula realiza a translação do mapa.

ii. Fluxos Alternativos

Igual o caso de uso “UC001 - Edição de Mapas Topográficos”.

iii. Pré-Condições

Igual o caso de uso “UC001 - Edição de Mapas Topográficos”.

iv. Condições Posteriores

Igual o caso de uso “UC001 - Edição de Mapas Topográficos”.

v. Pontos de Inclusão

Não se aplica.

vi. Pontos de Extensão

Não se aplica.

vii. Requisitos Especiais

Não se aplica.

6 Referências

Descreve-se aqui a informação necessária para que todas as fontes de dados citadas na ERSw possam ser recuperadas, caso necessário

A lista de referências deve ser completa e estar em acordo com as normas bibliográficas vigentes. Para cada referências é importante indicar, no mínimo, os autores, o título, o nome do veículo onde a referência foi divulgada e o ano da publicação.

ANEXO I

TEMPLATE PLANO DE TESTE

<Nome do Projeto>	Versão<1.0>
Plano de Testes	Data: <dd/mm/aaaa>
<identificador do documento>	



Histórico da Revisão

Dia	Autor	Versão	Descrição
<dd/mm/aaaa>	<autor>	<1.0>	<detalhes>

Equipe de Testes

Função	Responsabilidades
Desenvolvedor	Instalação e configuração do sistema para os testes; assessoria ao testador na especificação dos testes; assessoria à elaboração do relatório dos testes.
Coordenador de Equipe	Instalação e configuração do sistema para os testes; assessoria ao testador na especificação dos testes; assessoria à elaboração do relatório dos testes.
Analista de Testes	Planejamento dos testes; especificação dos testes; realização dos testes; assessoria à elaboração do relatório dos testes de integração.

Casos de Uso	Ação do Usuário	Resultado Esperado	Conforme		Incidentes	Período
			Sim	Não		
Transladar Mapa via Teclado	Clicar tecla W	Mover o mapa para cima e atualizar as coordenadas na Barra de Status	x			03/10/2009
	Clicar tecla S	Mover o mapa para baixo e atualizar as coordenadas na Barra de Status	x			03/10/2009
	Clicar tecla A	Mover o mapa para esquerda e atualizar as coordenadas na Barra de Status	x			03/10/2009
	Clicar tecla D	Mover o mapa para direita e atualizar as coordenadas na Barra de Status	x			03/10/2009
Edição de Mapa Topográfico	Selecionar Mapa	Apresentar o mapa selecionado na janela principal	x			03/10/2009
	Selecionar Transformação Geométrica	Cursos do mouse reflete a transformação selecionada	x			03/10/2009
	Selecionar Ponto de Origem	O ponto de origem é desenhado na Janela Principal	x			03/10/2009
	Mover o cursos	Os parâmetros da transformação selecionada são desenhados na Janela Principal	x			03/10/2009
	Selecionar Segundo Ponto	Apresentar Mapa Transformado. Mouse reflete estado ocioso do sistema.	x			03/10/2009

Interfaces	Ação do Usuário	Resultado Esperado	Conforme		Incidentes	Período
			Sim	Não		
	Clicar o botão esquerdo do mouse no botão "Abrir"	Abre Caixa de Dialogo “Abrir Arquivo”	x			03/10/2009
	Clicar o botão esquerdo do mouse no botão "Salvar"	Abre Caixa de Dialogo “Salvar Arquivo”	x			03/10/2009
	Clicar o botão esquerdo do mouse no botão "Transladar"	Cursos do mouse reflete a seleção. Botão “Transladar” pressionado, botões das demais transformações liberados.	x			03/10/2009
	Clicar o botão esquerdo do mouse no botão "Rotacionar"	Cursos do mouse reflete a seleção. Botão “Rotacionar” pressionado, botões das demais transformações liberados.	x			03/10/2009
Barra de Ferramentas	Clicar o botão esquerdo do mouse no botão "Escalar"	Cursos do mouse reflete a selecão. Botão “Escalar” pressionado, botões das demais transformações liberados.	x			03/10/2009

ANEXO I



<Nome do Projeto>

Relatório de Testes

Versão <1.0>

Histórico da Revisão

Data	Versão	Descrição	Autor
<dd/mmm/aa>	<x.x>	<detalhes>	<nome>

Índice

1. Objetivo 49
2. Descricão do Produto 49
3. Definições e Siglas 49
4. Configuração dos Testes 50
5. Participantes dos Testes 50
6. Atividades e Eventos 50
7. Incidentes 52
8. Variações 52
9. Abrangência 52
10. Sumário dos Resultados 52
11. Avaliação 52
12. Referências 52

Relatório de Testes

1 Objetivo

Descreve-se aqui o propósito desse documento.

Este relatório final tem por objetivo confrontar os requisitos levantados e a implementação do sistema de forma a detectar alguma falha, visando garantir a qualidade do software. No decorrer deste documento serão relatados os resultados dos testes realizados no sistema, nos quais a implementação de todos seus casos de uso foram analisados. Cada caso de uso gera uma especificação de teste funcional. Testes adicionais de caixa branca são usados para verificar se as interfaces.

2 Descrição do Produto

Faz-se aqui uma breve descrição sobre o produto, o contexto no qual ele se insere, e sobre o histórico do projeto.

O objetivo dessa seção de texto é tonar mais fácil o entendimento do documento e situar o leitor.

3 Definições e Siglas

Descreve-se aqui a definição de todas as siglas, abreviações e termos usados no documento.

Deve-se supor que este documento será lido tanto por desenvolvedores, quanto por usuários. Por isso o documento deve conter as definições relevantes tanto para os termos do domínio de aplicação, quanto para termos técnicos utilizados e que não são de conhecimento do público geral.

Tabela 3: Definições e Siglas

Sigla ou Termo	Descrição
-	-
-	-
-	-
-	-

4 Configuração dos Testes

N°	Hardware	Software	Ferramentas de teste	Componentes de teste
1	Proc: Core 2 Duo 6400 Veloc.: 2,13 GHz Mem. RAM: 3,25GB Pl. Vídeo: ATI Radeon 4560, 512 MB.	Windows XP SP2	Teste manual	-
2	Proc: Core 2 Duo E7300 Veloc.: 2,66 GHz Mem. RAM: 3,5GB Pl. Vídeo: NVidia GeForce 9600 GSO, 512 MB.	Windows XP SP2	Teste manual	-

5 Participantes dos Testes

Função	Responsabilidades
Programador Júnior	Instalação e configuração do sistema para os testes; assessoria ao testador na especificação dos testes; assessoria à elaboração do relatório dos testes.
Programador Sênior	Instalação e configuração do sistema para os testes; assessoria ao testador na especificação dos testes; assessoria à elaboração do relatório dos testes.
Analista de Sistema	Planejamento dos testes; especificação dos testes; realização dos testes; assessoria à elaboração do relatório dos testes de integração.

6 Atividades e Eventos

Casos de Uso/Interface	Ação do Usuário	Resultado Esperado	Conforme		Incidentes	Período
			Sim	Não		
Transladar Mapa via Teclado	Clicar tecla W	Mover o mapa para cima e atualizar as coordenadas na Barra de Status	x			10/03/2009
	Clicar tecla S	Mover o mapa para baixo e atualizar as coordenadas na Barra de Status	x			10/03/2009
	Clicar tecla A	Mover o mapa para esquerda e atualizar as coordenadas na Barra de Status	x			10/03/2009
	Clicar tecla D	Mover o mapa para direita e atualizar as coordenadas na Barra de Status	x			10/03/2009
Edição de Mapa Topográfico	Selecionar Mapa	Apresentar o mapa selecionado na janela principal	x			10/03/2009
	Selecionar Transformação Geométrica	Cursos do mouse reflete a transformação selecionada	x			10/03/2009

	Selecionar Ponto de Origem	O ponto de origem é desenhado na Janela Principal	x			10/03/2009
	Mover o cursos	Os parâmetros da transformação selecionada são desenhados na Janela Principal	x			10/03/2009
	Selecionar Segundo Ponto	Apresentar Mapa Transformado. Mouse reflete estado ocioso do sistema.	x			10/03/2009
Barra de Ferramentas	Clicar o botão esquerdo do mouse no botão "Abrir"	Abre Caixa de Dialogo “Abrir Arquivo”	x			10/03/2009
	Clicar o botão esquerdo do mouse no botão "Salvar"	Abre Caixa de Dialogo “Salvar Arquivo”	x			10/03/2009
	Clicar o botão esquerdo do mouse no botão "Transladar"	Cursos do mouse reflete a seleção. Botão “Transladar” pressionado, botões das demais transformações liberados.	x			10/03/2009
	Clicar o botão esquerdo do mouse no botão "Rotacionar”	Cursos do mouse reflete a selecão. Botão “Rotacionar” pressionado, botões das demais transformações liberados.	x			10/03/2009
	Clicar o botão esquerdo do mouse no botão "Escalar”.	Cursos do mouse reflete a selecão. Botão “Escalar” pressionado, botões das demais transformações liberados.	x			10/03/2009

7 Incidentes

Nenhum incidente detectado.

8 Variações

Não aplicáveis.

9 Abrangência

A cobertura pode ser considerada satisfatória dentro do previsto para estes testes do Módulo de Imersão Virtual.

10 Sumário dos Resultados

O produto passou nos testes, com correção de alguns bugs de implementação.

11 Avaliação

Levando-se em conta os resultados alcançados, a bateria de testes pode ser considerada de cobertura satisfatória.

12 Referências

Descreve-se aqui a informação necessária para que todas as fontes de dados citadas na ERSw possam
ser recuperadas, caso necessário

A lista de referências deve ser completa e estar em acordo com as normas bibligráficas vigentes. Para cada referências é importante indicar, no mínimo, os autores, o título, o nome do veículo onde a referência foi divulgada e o ano da publicação.

ANEXO I - ARTEFATOS DE SOFTWARE

Relatório de Acompanhamento Semanal



Igor Muzetti Pereira
igormuzetti@gmail.com

Tiago Garcia de Senna Carneiro
tiago@iceb.ufop.br

Departamento de Computação
Universidade Federal de Ouro Preto



Laboratório Associado INPE/UFOP para Modelagem e Simulação de Sistemas Terrestres

---

Preparado por: Parrera
Equipe: TerraLAB-2011
Data Data da semana em curso

Visão geral da semana anterior do cronograma (NetProject):

Projeto	Status	% Concluído	Início Planejado	Término Planejado	Início Real	Término Real	Prazo Estimado	Prazo Real
Nome Projeto	Em execução	0,00%	19/09	25/11	21/09		50	Em execução
+Etapa	Em execução	0,00%	19/09	25/11	21/09		50	Em execução

Visão geral atual do cronograma (NetProject):

Projeto	Status	% Concluído	Início Planejado	Término Planejado	Início Real	Término Real	Prazo Estimado	Prazo Real
Nome Projeto	Em execução	10,00%	19/09	25/11	21/09		50	Em execução
+Etapa	Em execução	10,00%	19/09	25/11	21/09		50	Em execução

Status dos Tickets(TRAC):

Verde (Ticket - Realizações, Progressos)
Os itens abaixo representam as tarefas(tickets) e progressos realizados na semana.

Responsável	Realizações	Data de realização da tarefa
Nome Colaborador		*N/A*

Amarelo (Ticket - Reaberto)
Os itens abaixo representam tarefas(tickets) que foram reabertas, ou seja, tarefas que irão ultrapassar a data planejada e terão que ter prazos estendidos.

Responsável	Ticket/Observação	1 Próxima data estimada
Nome Colaborador	*Ticket:* *Observação:*	*N/A*
	Ticket: *Observação:*	*N/A*

Vermelho (Ticket - Pendentes)
Os itens abaixo representam tarefas pendentes que foram planejadas porém ainda não foram concluídas. Estes itens podem incluir atividades/tarefas novas e/ou pendentes.

Responsável	Ticket	Motivo
Nome Colaborador	*Ticket:*	*N/A*
	Ticket:	*N/A*

Horas Trabalhadas (dd/mm até dd/mm):

Colaborador	Dias trabalhados	Horas trabalhadas
<nome>	*x*	*xx:xx:xx*
<nome>	*x*	*xx:xx:xx*