CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DO RIO GRANDE DO NORTE
DIRETORIA DE ENSINO
DEPARTAMENTO ACADÊMICO DE TECNOLOGIA DA INFORMAÇÃO
TECNOLOGIA EM DESENVOLVIMENTO DE SOFTWARE

BRUNO PEREIRA PONTES

SIGES: UM ESTUDO DE CASO APLICANDO O PROCESSO ACADÊMICO SIMPLIFICADO (PAS)

NATAL / RN
2007

BRUNO PEREIRA PONTES

SIGES: UM ESTUDO DE CASO APLICANDO O PROCESSO ACADÊMICO SIMPLIFICADO (PAS)

Monografia apresentada à Banca Examinadora do Trabalho de Conclusão do Curso de Tecnologia em Desenvolvimento de Software, em cumprimento às exigências legais como requisito parcial à obtenção do título de Tecnólogo em Desenvolvimento de Software.

Orientador: MSc, Leonardo Ataíde Minora
Co-Orientador: MSc, Fellipe Araújo Aleixo

NATAL / RN
2007

Divisão de Serviços Técnicos.
Catalogação da publicação na fonte.
CEFET/RN. Biblioteca Sebastião Fernandes

Pontes, Bruno Pereira.

SIGES: Um estudo de caso aplicando o Processo Acadêmico Simplificado (PAS) / Bruno Pereira Pontes. Natal / RN, 2007.

63f.

Orientador: Leonardo Ataíde Minora.

Co-orientador: Fellipe Araújo Aleixo.

Monografia (Graduação de Tecnologia em Desenvolvimento de Software) – Centro Federal de Educação Tecnológica do Rio Grande do Norte.

1. Engenharia de software – Monografia. 2. Processo de software – Monografia. 3. Desenvolvimento de software – Monografia. I. Minora, Leonardo Ataíde. II. Aleixo, Fellipe Araújo. III. Título.

CEFET/RN/BSF

BRUNO PEREIRA PONTES

SIGES: UM ESTUDO DE CASO APLICANDO O PROCESSO ACADÊMICO SIMPLIFICADO (PAS)

Monografia apresentada à Banca Examinadora do Trabalho de Conclusão do Curso de Tecnologia em Desenvolvimento de Software, em cumprimento às exigências legais como requisito parcial à obtenção do título de Tecnólogo em Desenvolvimento de Software.

Aprovada em _____ de _____ de 200 ___

Apresenta à comissão examinadora integrada pelos seguintes membros:

_____

Leonardo Ataíde Minora, MSc.
Orientador - CEFET/RN

_____

Fellipe Araújo Aleixo, MSc.
Membro Examinador - CEFET/RN

_____

Alessandra Mendes Pacheco Nascimento, MSc.
Membro Examinador - CEFET/RN

Dedico este trabalho a meus pais,
cujo exemplo de honestidade e trabalho
tem sido um norteador para a minha vida,
e para minha futura esposa, que tem
me dado apoio nos momentos mais difíceis
e mostrado a simplicidade de ter esperança.

AGRADECIMENTOS

Dedico meus sinceros agradecimentos para: minha Mãe e minha irmã Zepa por terem suportado tantas madrugadas de barulho e garagem cheia de carros (Amo Vocês); meu Pai: por ter entendido e aceitado a minha ausência em vários momentos nesses últimos 03 anos (Te Amo Cara); Leo e Guga pelas vitórias, conquistas, apoio, momentos difíceis e as várias noites sem dormir; ao pai, mãe e irmã de Guga por terem nos aguentado durante intermináveis madrugadas de estudo; Nathalie pelo amor, apoio, incentivo, e também por ter suportado minha chatisse e minha ausência (Te Amo); Shanda pelo apoio e incentivo prestados no início dessa jornada; Tio Minora por todas as orientações e pelos ensinamentos passados; Bráulio pelo trabalho vitorioso realizado no Nuarq; Fellipão por ter aberto as portas para este trabalho; Robinson, Barão, Teobaldo, Ribamar e Valetim pela competência, profissionalismo e ensinamentos passados, pois estão sendo bastante úteis; a equipe 5 homens e 1 segredo por terem encarado o desafio de desenvolver o SIGES; ao Sgt Fabricio pelo braço amigo, nos momentos difíceis; ao Cap Alzeir pela competência e motivação para o estudo; aos integrantes da Seção de Informática por todo o apoio prestado nos momentos mais difícies; Dona Salete pelo carinho e atenção que sempre demonstrou; a Família Cefet pelo acolhimento e amizades conquistadas; e por fim a família Itapiru pela confiança no meu trabalho.

“A educação faz um povo fácil de ser liderado, mas difícil de ser dirigido;
fácil de ser governado, mas impossível de ser escravizado!!!”

Henry Peter

RESUMO

Atualmente, os cursos voltados ao desenvolvimento de sistemas computacionais não possuem um processo de software que atenda às suas necessidades. Processos convencionais, também conhecidos como prescritivos, como o Rational Unified Process (RUP), baseado no Processo Unificado, são complexos e burocráticos, o que dificulta o aprendizado do discente. Viu-se, então, a necessidade de um processo didático para se utilizar no meio acadêmico, que permita aos discentes entenderem as etapas de desenvolvimento de um sistema computacional, aplicando, na prática, um processo mais leve, mas que abranja todas as fases do seu ciclo de vida. Os alunos do curso superior de Tecnologia em Desenvolvimento de Software, do Centro Federal de Educação Tecnológica do Rio Grande do Norte (CEFET-RN), desenvolveram o Processo Acadêmico Simplificado (PAS), com o intuito de torná-lo o processo de software base nas disciplinas de Engenharia de Software, facilitando o aprendizado do discente. Quando da construção do PAS, no entanto, não houve tempo hábil para aplicá-lo a nenhum projeto de desenvolvimento de software. Este trabalho tem como objetivo aplicar o PAS no desenvolvimento do Sistema de Gerência de Escalas de Serviço (SIGES), destinado às Organizações Militares (OM) do Exército Brasileiro.

Palavras-chave: Engenharia de Software. Processo de Software. Desenvolvimento de Software.

ABSTRACT

Now a days, the Graduate Courses about Software Development don't have a Software Development Process that supplies students requirements. Prescriptives processes, like Unified Process and others, has high complexity and are very bureaucratic and students have difficulty to learn. Because this, a new Process was construct and its objective was a pratice of software development and its phases in Academic Environment. This pratice must be low complexity and low bureaucratic and it promote the execution of phases and life cycle of the development softwares. The students of the Tecnologia em Desenvolvimento de Software Graduation Course, of the Centro Federal de Educação Tecnologica do Rio Grande do Norte (CEFET-RN), construct the process named Processo Acadêmico Unificado - PAS. The PAS intention is to be the process of the software engineering courses and making student learning easear. However, when the PAS was constructed doesn't have time to apply in the case study of software development project. In this work, the PAS is applied in case study named Sistema de Gerência de Escala de Serviço do Exército Brasileiro - SIGES, and the its result is presented.

Key words: Software Engineering. Software Process. Software Development.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CEFET-RN	– Centro Federal de Educação Tecnológica do Rio Grande do Norte
PAS	– Processo Acadêmico Simplificado
SCAMPI	– Standard CMMI Appraisal Method for Process Improvement
OM	– Organização Militar
SIGES	– Sistema de Gerência de Escalas de Serviço
PMBOK	– Project Managment Body of Knowledge Guide
TDS	– Tecnologia em Desenvolvimento de Software
XP	– eXtreme Programming
UML	– Unified Modeling Language
RUP	– Rational Unified Process
IDE	– Integrated Development Environment
SGBD	– Sistema de Gerenciamento de Bancos de Dados
JSF	– JavaServer Faces
JSTL	– JavaServer Pages Standard Tag Library
EJB3	– Enterprise JavaBeansTM 3.0
JPA	– Java Persistence API
ANT	– Another Neat Tool

SUMÁRIO

1 INTRODUÇÃO

Durante o período de formação acadêmica, os alunos dos cursos voltados ao desenvolvimento de sistemas computacionais recebem uma grande carga de conhecimento na área da Engenharia de Software. Tal conhecimento é posto em prática nas disciplinas de desenvolvimento de sistemas, proporcionando ao discente uma visão real do trabalho com a Engenharia de Software. Durante a realização de tais disciplinas, os alunos utilizam um conjunto de ferramentas, métodos e práticas, com o objetivo de construir um sistema. Esse conjunto é denominado de Processo de Software (HUMPHREY, 1989).

Esse ambiente acadêmico, no entanto, apresenta características próprias, como a limitação de recursos e a inexperiência dos estudantes, que causam impacto no processo utilizado, tornando o mesmo inadequado e ineficaz, fazendo com que o aproveitamento do discente fique aquém do desejado. A partir de experiências empíricas realizadas em sala de aula no Curso de Tecnologia em Desenvolvimento de Software no CEFET-RN, verificou-se que não existe um processo que atenda as necessidades do meio acadêmico, pois os processos definidos na literatura são complexos e burocráticos demais, impossibilitando a sua execução por completo, tornando o aprendizado do aluno deficiente. Tornou-se necessário então, a definição de um processo que pudesse atender as peculiaridades do contexto acadêmico.

Todavia, essa definição não acontece de forma universal (FALBO, 2000). Para ser eficaz e conduzir a construção de produtos de boa qualidade, o processo deve ser adequado ao domínio da aplicação e ao projeto específico. Desse modo, processos devem ser definidos caso a caso, considerando-se, principalmente, a organização onde o produto será desenvolvido e o grupo de desenvolvimento, fornecendo assim, estabilidade, controle e organização para atividades que, se deixadas sem controle, poderiam tornar-se bastante caóticas (SOMMERVILLE, 2003; PRESSMAN, 2006; INSTITUTE, 2004).

Garcia (GARCIA et al., 2004) propõem que, para que um processo seja satisfatoriamente utilizado em ambiente acadêmico, ele deve:

- ser adequado às características dos projetos desenvolvidos, que possuem pequeno escopo; valorizar avaliações contínuas e iterativas com a presença de clientes reais e usar tecnologias consideradas estado da arte;
- focar na aprendizagem do processo com alguns elementos qualificadores: boa produtividade, geração mínima de artefatos, etc;

- ser de fácil entendimento e simples, mas robusto e completo o suficiente para gerar produtos de qualidade.

Diante desse cenário, foi desenvolvido um Processo de Software denominado de Processo Acadêmico Simplificado (PAS) (PONTES; ALEIXO; MINORA, 2006), com o intuito de facilitar o aprendizado de processo de software nos cursos da área de desenvolvimento de sistemas. A intenção do desenvolvimento foi a especificação de um processo de software com as seguintes características: simples, minimamente formal e ágil, possibilitando a experiência acadêmica de construção de software orientado a um processo.

1.1 OBJETIVOS DO TRABALHO

1.1.1 OBJETIVOS GERAIS

O objetivo geral deste trabalho é o relato da construção de um sistema computacional, utilizando o Processo Acadêmico Simplificado, possibilitando aos discentes a prática dos conhecimentos adquiridos nas diciplinas de Engenharia de Software.

1.1.2 OBJETIVOS ESPECÍFICOS

Os objetivos específicos deste trabalho são:

- Realizar um estudo sobre os conceitos básicos relativos a processos de desenvolvimento de software;
- Conhecer os processos mais utilizados no desenvolvimento de sistemas computacionais;
- Executar o acompanhamento da fase de concepção do PAS;
- Executar o acompanhamento da fase de elaboração do PAS;
- Executar o acompanhamento da fase de construção do PAS;
- Executar o acompanhamento da fase de validação do PAS;
- Documentar o desenvolvimento e resultados do projeto

1.2 JUSTIFICATIVA

A aplicação de um processo de software não é garantia de que o software será entregue no prazo, satisfazendo às necessidades do cliente; ou de que ele exiba as características técnicas

que levarão a qualidade no longo prazo (PRESSMAN, 2006). Para que se consiga satisfazer a um conjunto de critérios básicos, que demostraram ser essenciais para uma engenharia de software bem sucedida, o processo de software deve ser continuamente avaliado. A relação entre os métodos de avaliação e melhoria e o processo de software podem ser vistos na figura 1.1.

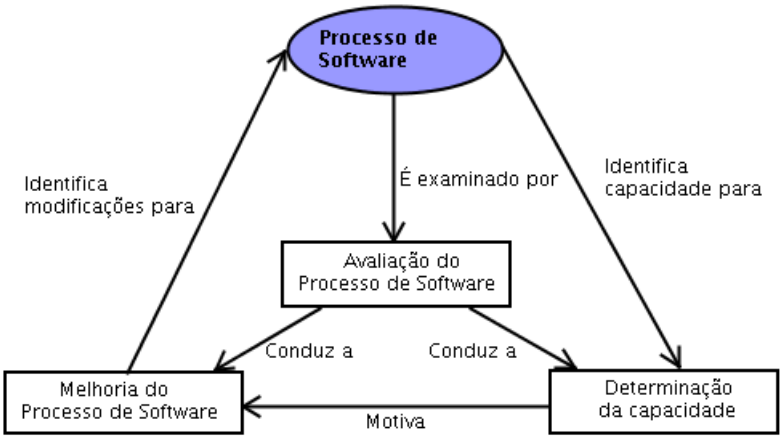

Figura 1.1: Relação entre Processo de Software e Métodos de Avaliação.
Fonte: Extraída de (PRESSMAN, 2006).

Diferentes abordagens de avaliação de processo de software têm sido propostas durante as últimas décadas (PRESSMAN, 2006). O Standard CMMI Appraisal Method for Process Improvement (SCAMPI) realiza uma avalição com base na verificação dos Indicadores de Implementação de Prática (*Practice Implementation Indicators* – PII), representados por artefatos diretos e indiretos produzidos pela execução do processo, ou por afirmações da organização avaliada, que equivalem a artefatos indiretos (ITABORAHY et al., 2005). Seja qual for a abordagem utilizada, é essencial que o processo seja aplicado a algum projeto de desenvolvimento de software, para que, apartir daí, possa ser avaliado e, se for o caso, corrigido.

1.3 METODOLOGIA

A execução do estudo de caso ocorreu na disciplina de Prática de Desenvolvimento de Software I (PDS-I) do Curso de Técnologia em Desenvolvimento de Software do CEFET-RN, no período 2007.1, com carga horária de 120h, sendo dividida em três etapas distintas: (i) escolha do sistema a ser desenvolvido (cliente), (ii) escolha da equipe de desenvolvimento e (iii) construção do sistema.

A primeira etapa foi realizada antes mesmo do início do semestre letivo. Em uma reunião entre o professor responsável pela disciplina de PDS-I e o cliente do sistema, definiu-se que seria

desenvolvido o Sistema de Gerência de Escalas de Serviço (SIGES), destinado às Organizações Militares (OM) do Exército Brasileiro. Nessa mesma reunião foi definido o escopo do sistema. Três fatores foram essenciais para essa escolha: o tamanho do sistema, o seu nível de complexidade e o próprio cliente. O tamanho e a complexidade necessitavam ser suficientes para que o processo fosse aplicado, por completo, dentro do tempo disponível. Preferencialmente, o cliente deveria existir de fato, possibilitando a implantação do sistema em um ambiente de real utilização.

Tendo o sistema sido escolhido e seu escopo definido, foi escolhida a equipe que realizaria o desenvolvimento do mesmo. A escolha deu-se no primeiro dia de aula do semestre letivo 2007.1, acontecendo de maneira voluntária. Na mesma etapa, foi apresentado à equipe o representante do cliente, que serviria de homem de ligação.

Logo após definida a equipe, foi iniciada a construção do sistema, com a execução das primeiras atividades da fase de concepção, fase inicial do PAS. O processo foi sendo apresentado a equipe, em reuniões semanais, à medida que o mesmo ia acontecendo. Tais reuniões foram sempre registradas em atas e as mesmas disponibilizadas na página do sistema[1]. Ao final de cada fase, foi realizado um fechamento. Todos os documentos gerados, serviram como registro da aplicação do processo, sendo a base da elaboração deste trabalho.

1.4 ORGANIZAÇÃO DO TRABALHO

O trabalho está organizado da seguinte maneira: O segundo capítulo aborda os conceitos básicos a respeito de processo de software. O terceiro capítulo apresenta o Processo Acadêmico Simplificado, utilizado no desenvolvimento do SIGES . O quarto capítulo relata a aplicação do PAS na construção do SIGES, abordando todas as etapas do seu desenvolvimento. Por fim, o quinto capítulo trata da conclusão e trabalhos futuros.

[1] http://trac.itapirunet.com.br/wiki/siges

2 PROCESSO DE SOFTWARE

Para Baetjer (BAETJER, 1997), o desenvolvimento de software é um processo de aprendizado social, que ocorre de maneira iterativa, no qual a própria ferramenta serve como meio de comunicação, com cada nova rodada de diálogo atraindo mais conhecimento útil do pessoal envolvido. O resultado desse processo iterativo de aprendizado é uma incorporação de conhecimentos coletados, destilados e organizados, a medida que o processo é conduzido (PRESSMAN, 2006), formando ao final do mesmo um produto de software.

Sommerville (SOMMERVILLE, 2003) define esse processo como sendo um conjunto estruturado de atividades e resultados associados que produzem o produto de software. Para ele, existem 4 (quatro) atividades básicas comuns a qualquer processo: (i) especificação do software, (ii) projeto e implementação de software, (iii) validação do software e (iv) evolução do software.

Para Pressman (PRESSMAN, 2006), o processo de software é definido como sendo um framework (estrutura de suporte) para as tarefas que são necessárias para a construção de software de alta qualidade. Ele define 5 (cinco) atividades genéricas desse framework que podem ser aplicadas a maioria dos projetos de software: (i) comunicação, (ii) planejamento, (iii) modelagem, (iv) construção e (v) implantação.

A partir dessas definições, considerada-se neste trabalho que, de forma geral, um processo de software pode ser visto como um conjunto de atividades, métodos, ferramentas e práticas que são utilizadas para construir um produto de software. Logo, quando se define um processo de software, devem ser consideradas as seguintes informações: atividades a serem realizadas, recursos necessários, artefatos requeridos e produzidos, procedimentos adotados e o modelo de ciclo de vida utilizado (FALBO, 1998).

É importante salientar que não existe processo correto ou incorreto; dependendo da sua aplicação, ambiente e objetivo, o uso de um processo específico pode ser vantajoso ou não. Um ponto importante a ressaltar é que, como cada autor classifica processos e seus elementos básicos de forma diferente, torna difícil uma uniformidade completa. As seções seguintes definem esses elementos básicos, presentes em qualquer processo de software, e se baseiam em características comuns encontradas na literatura.

2.1 CONCEITOS BÁSICOS

2.1.1 CICLO DE VIDA

O PMBOK (INSTITUTE, 2004) define projeto como um esforço temporário empreendido para criar um produto, serviço ou resultado exclusivo. O termo temporário significa que todos os projetos possuem um início e um final definidos. O ciclo de vida do desenvolvimento de um software define, basicamente, as fases que conectam esse início e fim, considerando os seguintes aspectos (INSTITUTE, 2004):

- Que trabalho técnico deve ser realizado em cada fase;
- Quando as entregas devem ser geradas em cada fase e como cada entrega é revisada, verificada e avaliada;
- Quem está envolvido em cada fase;
- Como controlar e aprovar cada fase.

A transição entre as fases do ciclo de vida normalmente é definida por alguma forma de transferência técnica ou entrega. As entregas de uma fase geralmente são revisadas, para garantir que estejam completas e exatas, e aprovadas antes que a próxima fase seja iniciada. No entanto, não é incomum que uma fase seja iniciada antes da aprovação das entregas da fase anterior, quando os riscos envolvidos são considerados aceitáveis. Embora muitos ciclos de vida possuam nomes de fases semelhantes com entregas semelhantes, poucos são idênticos. Alguns podem ter quatro ou cinco fases, mas outros podem ter nove ou mais (INSTITUTE, 2004).

Existem alguns modelos teóricos desenvolvidos, conhecidos como modelos de ciclo de vida ou modelos de processos, que buscam descrever a forma com que as fases seguem e interagem. Eles são uma filosofia do andamento das fases, e não uma descrição de como cada atividade deve ser executada. Os processos de software são descritos através desses modelos, a partir de uma pespectiva específica, sendo úteis para explicar diferentes abordagens do desenvolvimento de software (SOMMERVILLE, 2003).

Alguns dos modelos mais conhecidos são: modelo cascata, modelo espiral, modelo protótipo e modelo iterativo e incremental. Na seção 2.2 o modelo iterativo e incremental será detalhado, já que o Processo Acadêmico Simplificao utiliza esse modelo de ciclo de vida.

2.1.2 FASES E DISCIPLINAS

Scott (SCOTT, 2003) define uma disciplina como sendo uma coleção de atividades relacionadas a uma "área de interesse" principal. Diferentes modelos de processos organizam essas disciplinas de maneiras diversas (AMBLER, 2004). No entanto, existe um conjunto delas que está presente em todos eles. Segundo o PMBOK (INSTITUTE, 2004), as disciplinas básicas de todo e qualquer processo são: (i) entender as necessidades do cliente, (ii) planejar a solução, (iii) implementar a solução, (iv) validar esta solução e (v) entregar o produto ao cliente. Tais atividades podem ser reescritas usando termos da Engenharia de Software como atividades de: (i) requisito, (ii) análise e projeto, (iii) implementação e (iv) teste (SOMMERVILLE, 2003).

A (i) disciplina de requisitos é responsável pela obtenção de um conjunto de requisitos de um produto de software (sejam eles funcionais ou não-funcionais) que devem ser acordados entre cliente e desenvolvedor. É ela quem ajuda o desenvolvedor a entender e delimitar o escopo e as funcionalidades do sistema a ser desenvolvido. A (ii) disciplina de análise e projeto visa o detalhamento, a estruturação do sistema e a validação dos requisitos que foram eliciados durante a disciplina de requisitos. Essa disciplina dá forma ao domínio do sistema, gerando um modelo conceitual do problema. Na (iii) disciplina de implementação o projeto elaborado na disciplina anterior, é, de fato, concretizado em uma linguagem de programação. Por fim, a (iv) disciplina de teste verifica se existem erros na implementação, bem como o comportamento adequado a nível das funções e módulos básicos do sistema.

Após conhecer as disciplinas básicas de um processo de software, é necessário entender como elas acontecem dentro deste. Essa organização é definida pelas fases (miniprojetos) do ciclo de vida de desenvolvimento. Para cada miniprojeto, as disciplinas que serão executadas são ordenadas para que os objetivos da fase sejam alcançados (SCOTT, 2003; INSTITUTE, 2004). O Processo Unificado (SCOTT, 2003), apresenta as disciplinas distribuídas em quatro fases: concepção, elaboração, construção e transição.

2.1.3 MODELOS, DOCUMENTOS E ARTEFATOS

Um artefato é um pedaço significativo de informação que é produzido, modificado ou utilizado em um processo. Os artefatos são os produtos de um projeto, ou seja, o que é produzido durante o seu desenvolvimento (SCOTT, 2003). Artefatos são utilizados como entradas de disciplinas e são produzidos como saída. Os artefatos podem ter várias formas:

- Um modelo, como um modelo de caso de uso, um modelo de projeto;
- Um elemento de um modelo, como uma classe, um caso de uso, um sub-sistema;

- Um documento, como um caso de negócio, glossário, visão;
- Código fonte;

É importante ressaltar a diferença existente entre modelo e documento. Um modelo é uma abstração que descreve um ou mais aspectos de um problema ou uma solução possível para ele (AMBLER, 2004). Já o documento é um artefato produzido por uma ferramenta de processamento de texto para fins de documentação dos principais aspectos de engenharia de um projeto, incluindo aspectos selecionados dos modelos e aspectos não-modeláveis (FILHO, 2003).

2.1.4 TRABALHADORES, PARTICIPANTES OU PAPÉIS

Um papel (trabalhador ou participante) define o comportamento e as responsabilidades de um determinado indivíduo ou grupo de indivíduos trabalhando como uma equipe. Papéis não são indivíduos e nem títulos de trabalho (SCOTT, 2003). Um indivíduo pode assumir vários papéis durante o desenvolvimento de um sistema. São exemplos de papéis:

- Analista de sistema – responsável por coordenar a obtenção dos requisitos e a modelagem dos casos de uso identificando funcionalidades do sistema e estabelecendo limites do sistema;
- Projetista – define responsabilidades, operações, atributos, relacionamentos de uma ou mais classes e determina como elas devem ser ajustadas para serem implementadas no ambiente;
- Projetista de testes – responsável pelo planejamento, projeto, implantação e avaliação de testes, incluindo a geração de plano e modelo de teste, implementando procedimentos de testes e avaliando a abrangência dos testes, resultados e a efetividade.
- Cliente - pessoa ou organização que utilizará o produto do projeto.

Quando um indivíduo está exercendo um determinado papel, produzindo um resultado importante para o contexto do projeto, ou seja, um artefato, está na verdade executando uma unidade de trabalho, chamada de atividade (SCOTT, 2003). Cada atividade pode ser dividida em passos e um conjunto dessas atividades forma uma disciplina, conforme dito na seção 2.1.2. A seguir alguns exemplos de atividades:

- Planejar uma iteração: realizada pelo papel gerente de projeto;

- Encontrar casos de uso e atores: realizada pelo papel analista de sistemas;
- Rever o projeto: realizada pelo papel revisor de projeto;
- Executar um teste de performance: realizado pelo papel testador de performance.

2.2 CLASSIFICAÇÃO DE PROCESSOS

2.2.1 SEGUNDO O CLICO DE VIDA

Sommerville (SOMMERVILLE, 2003) classifica os modelos de processo segundo seu ciclo de vida em: modelo em cascata, desenvolvimento evolucionário, desenvolvimento formal de sistemas, desenvolvimento orientado a reuso e modelos híbridos. Os modelos híbridos são o espiral e o iterativo e incremental. Será considerado neste trabalho apenas o modelo iterativo e incremental, pois o processo utilizado no estudo de caso está baseado em tal modelo.

2.2.2 SEGUNDO A FORMA DE TRABALHO

Quanto à forma de trabalho, os modelos de processos são classificado em prescritivos e ágeis (AMBLER, 2004). A seguir uma breve explicação sobre cada um deles.

Os modelos prescritivos definem um conjunto distinto de atividades, ações, tarefas, marcos e produtos de trabalho que são necessários para fazer engenharia de software com alta qualidade (PRESSMAN, 2006). Eles foram criados com o objetivo de por ordem no caos do desenvolvimento de software. Esses processos, no entanto, possuem uma deficiência importante: eles esquecem as fragilidades das pessoas que constroem software de computador (COCKBURN, 2002). Eles consideram que, utilizando-se o processo correto, no lugar correto e com um número necessário de artefatos, as pessoas participantes do projeto podem ser trocadas com facilidade (AMBLER, 2004). Apesar de não serem perfeitos, os modelos prescritivos fornecem um roteiro útil para o trabalho de engenharia de software.

Os modelos ágeis combinam uma filosofia e um conjunto de diretrizes de desenvolvimento. A filosofia encoraja a satisfação do cliente e a entrega incremental do software logo de início; equipes de projeto pequenas, altamente motivadas; métodos informais; produtos de trabalho de engenharia de software mínimos e simplicidade global de desenvolvimento. As diretrizes de desenvolvimento enfatizam a entrega em contraposição à análise e ao projeto (apesar dessas atividades não serem desencorajadas) e a comunicação ativa e contínua entre desenvolvedores e clientes (PRESSMAN, 2006). Em outras palavras, os processos ágeis valorizam indivíduos e interações, em vez de processos e ferramentas; software funcionando em vez de documentação

abrangente; colaboração do cliente, em vez de contratos; e repostas a modificações, em vez de seguir um plano (AMBLER, 2004).

2.2.3 MODELO ITERATIVO E INCREMENTAL

Iteração é um miniprojeto que resulta em uma versão do sistema, que contém funcionalidade adicionada ou melhorada em comparação com a versão anterior (incremento) (SCOTT, 2003; SOMMERVILLE, 2003; AMBLER, 2004). Em um processo de desenvolvimento incremental, a equipe, juntamente com o cliente, identifica as funções mais importantes e quais são menos importantes (AMBLER, 2004). Em seguida são definidas as iterações e as funções são alocadas em cada iteração de acordo com sua prioridade (AMBLER, 2004).

O modelo iterativo e incremental (Figura 2.1) proporciona vários benefícios, dentre os quais destacam-se (MARTINS, 1999): (i) redução dos riscos envolvendo custos a um único incremento; (ii) redução do risco de lançar o produto no mercado fora do cronograma previsto; (iii) aceleração do tempo de desenvolvimento do projeto como um todo; (iv) reconhecimento de uma realidade freqüentemente ignorada "mudanças de requisitos".

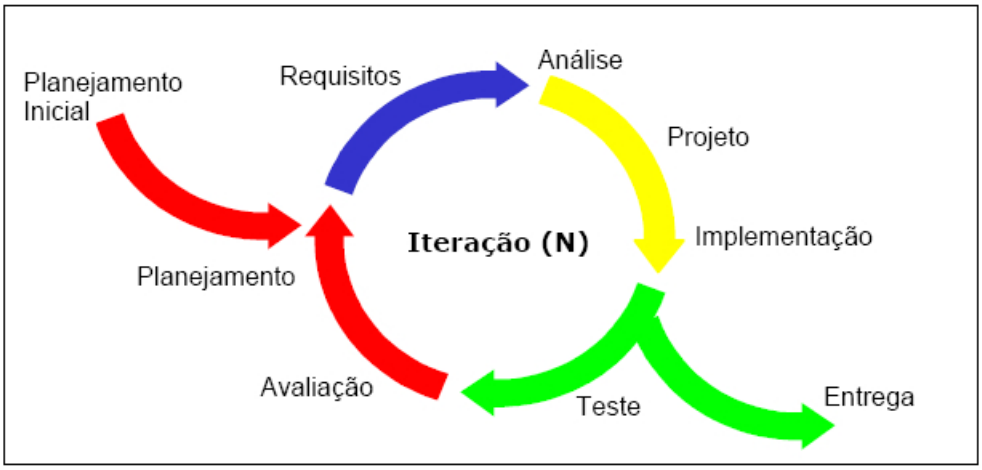

Figura 2.1: Modelo Iterativo e Incremental.
Fonte: Adaptado do Processo Unificado da Rational (RATIONAL SOFTWARE, 2007).

Esse modelo é emergente (AMBLER, 2004). Seu enfoque é interessante no sentido de que usa a flexibilidade e a modularidade do desenvolvimento orientado a objeto. Assim, provê um ciclo de vida que combina a preocupação de "como as pessoas trabalham" e concede o controle de gerenciamento (CANTOR, 1998). Essas características estão presentes em grande parte das metodologias atuais, que focam os aspectos práticos do desenvolvimento, e não em questões filosóficas profundas (AMBLER, 2004).

3 PAS (PROCESSO ACADÊMICO SIMPLIFICADO)

O PAS é um processo de desenvolvimento de software didático, desenvolvido para ser aplicado no meio acadêmico. Ele é baseado no modelo iterativo e incremental e possui as seguintes características: simples, minimamente formal e ágil; o que torna possível a experiência acadêmica de construção de software orientado a um processo, moldando-se as necessidades dos cursos voltados ao desenvolvimento de sistemas computacionais (PONTES; ALEIXO; MINORA, 2006).

O seu desenvolvimento ocorreu na disciplina de Processo de Desenvolvimento de Software do Curso de Tecnologia em Desenvolvimento de Software (TDS) do CEFET-RN. A disciplina não é regular do curso, mas foi oferecida no caráter de optativa no semestre 2006.2, transcorrendo em reuniões semanais de 3 horas, num total de 54 horas no semestre. É importante ressaltar que todos os alunos matriculados em tal disciplina já haviam cursado a disciplina de "Engenharia de Software I", no quarto período do curso. Nesta disciplina, o aluno adquiriu o conhecimento fundamental sobre Processos de Desenvolvimento de Software e conheceu os principais processos da literatura, como o Processo Unificado (SCOTT, 2003), Processo Unificado da Rational (JACOBSON; BOOCH; RUMBAUGH, 1999), eXtreme Programming (BECK, 2004), Scrum (SCHWABER, 2004), Crystal Clear (COCKBURN, 2005), Microsoft Solution Framework (MICROSOFT, 2007). Além dessa base teórica, os alunos puderam praticar o uso de um processo no desenvolvimento de software com uma pequena equipe (4 a 6 alunos por equipe gerenciada por professores) na disciplina de "Prática de Desenvolvimento de Software I", oferecida em 2006.1.

A metodologia utilizada para a construção do processo baseou-se na construção coletiva de conhecimento (SAVIANI, 2003). A cada aula, os alunos apresentavam o resultado de tarefas realizadas fora de sala de aula. Tais resultados eram discutidos e modificados para refletir todas as contribuições e reflexões realizadas no debate, sempre com a mediação do professor da disciplina. Cada tarefa realizada pelos alunos teve associada a si uma pontuação, utilizada para compor as médias parciais e finais dos alunos na disciplina.

No primeiro encontro da disciplina de Processo de Software, foi realizada uma revisão geral de todos os conceitos importantes relativos ao tema central da disciplina. No final do primeiro encontro, foram passadas as tarefas individuais para serem apresentadas na aula seguinte. As atividades iniciais da disciplina objetivaram definir a estrutura base para o Processo de Software proposto, suas fases e disciplinas, bem como um nome para o mesmo. Com a estrutura base do

processo definida, as atividades passadas para os alunos foram de três tipos: (i) descrever características e objetivos de fases, disciplinas e artefatos; (ii) proposição de modelos de artefatos; e (iii) proposição de fluxo de atividades para uma dada disciplina em uma dada fase.

Os encontros posteriores à primeira aula, foram divididos em dois momentos: no primeiro momento os alunos expunham os resultados das suas tarefas, e no segundo momento o grupo discutia o que foi apresentado. Na discussão, as propostas de melhoramento estavam sempre em pauta, a fim de chegar a um consenso que definisse as características específicas do processo proposto.

Todas as informações geradas pelos alunos foram publicadas em um sítio na internet[1], equipado com uma ferramenta simplificada de gerenciamento de conteúdo – DokuWiki[2]. A medida que os debates e definições foram acontecendo, as informações foram sendo atualizadas no sítio. Como resultado final, foi gerado um sítio contendo descrições, objetivos, fluxos de atividades e modelos de artefatos para cada disciplina em cada fase do processo.

As disciplinas definidas para o PAS foram: (i) Requisitos, (ii) Análise e Projeto e (iii) Implementação e Testes. Mesmo ocorrendo uma fusão de disciplina, no intuito de simplificar o processo, houve um atendimento as disciplinas básicas supra citadas no capítulo 2. Durante as reuniões onde foram definidas quais disciplinas deveriam fazer parte do PAS, foi apresentada uma proposta de inclusão da disciplina de Gerência de Projetos. Contudo, chegou-se a conclusão de que esta disciplina deveria fazer parte do processo em um segundo momento, após uma primeira aplicação do PAS, onde seriam observados todos os aspectos que deveriam compor a referida disciplina.

A (i) disciplina de Requisitos é responsável pela obtenção de um conjunto de requisitos de um produto de software (sejam eles funcionais ou não-funcionais) que devem ser acordados entre cliente e desenvolvedor. É ela quem ajuda o desenvolvedor a entender e delimitar o escopo e as funcionalidades do sistema a ser desenvolvido. Fornece uma base para planejar o conteúdo técnico e o gerenciamento das iterações e para estimar o custo e o tempo de desenvolvimento do sistema. Nela pode-se definir também um protótipo de interface de usuário para o sistema, focando nas necessidades e metas do usuário.

A (ii) disciplina de Análise e Projeto visa o detalhamento, a estruturação do sistema, além da validação dos requisitos que foram eliciados durante a disciplina de requisitos. Essa disciplina dá forma ao domínio do sistema, gerando um modelo conceitual do problema orientado pelos casos de uso.

[1] Temporariamente disponível em http://tads.cefetrn.br/pas

[2] Ferramenta de Wiki disponível em http://wili.splitbrain.org/

Na (iii) disciplina de Implementação e Testes, o projeto elaborado na disciplina anterior é, de fato, concretizado em uma linguagem de programação orientada a objetos. A relação entre essas duas disciplinas é bastante estreita, pois a implementação do projeto poderá levar a ajustes e adaptações no mesmo. Dentro dos aspectos a serem considerados na disciplina de Implementação e Testes, destaca-se o planejamento e execução de testes e a aplicação das seguintes práticas de Programação Extrema - XP: desenvolvimento orientado a testes, refatoração, integração contínua, código comunitário, padrão de codificação e releases pequenos (BECK, 2004).

Cada uma das disciplinas possui um fluxo de atividades para cada fase, definindo o que deve ser feito em cada uma dessas fases. Os fluxos são descritos em diagrama de atividades UML , onde as atividades são escritas em português declarativo.

Dessa maneira, o PAS pretende unir as virtudes dos processos prescritivos e ágeis. Ele é centrado em uma arquitetura baseada em componentes, oferece uma abordagem baseada em disciplinas, é guiado por casos de uso, seu ciclo de vida é iterativo e incremental, é projetado e documentado utilizando a notação UML, características herdadas do RUP . No entanto, ele é leve, eficiente, flexível e possui os valores da simplicidade, comunicação e feedback, definidos em processos ágeis.

3.1 DEFININDO O CICLO DE VIDA DO PAS

Foram definidas quatro fases para o PAS: Concepção, Elaboração, Construção e Validação. Cada fase define um marco do processo e estabelece um fluxo de atividades para cada disciplina, bem como os artefatos trabalhados (gerados ou modificados) por estas atividades. O final de cada fase é caracterizado pelo cumprimento dos objetivos pré-estabelecidos. A seguir, será detalhada cada fase, incluindo suas disciplinas com seus respectivos fluxos de atividades.

3.1.1 FASE DE CONCEPÇÃO

A fase de concepção busca uma visão inicial para se trabalhar o sistema, deixando como resultado, a clara visualização do sistema a ser desenvolvido (seu escopo, seus principais requisitos e desejos do cliente). Todas as atividades desenvolvidas nesta fase têm este fim, e para tanto necessitam da participação ativa do cliente. Trata-se de uma fase marcada por reuniões e discussões onde inicialmente o cliente apresenta sua visão do sistema, e num segundo momento os projetistas apresentam a visão que puderam capturar do mesmo.

A fase é dada por terminada não necessariamente tendo passado o tempo estipulado para a

mesma, mas tendo o cliente e projetistas concordados acerca do que é o sistema. No entanto, deve-se tomar muito cuidado com a duração desta fase, pois o prolongamento exacerbado ou a redução de sua duração poderá prejudicar as demais fases do projeto.

Ao final da fase, os seguintes objetivos devem ser atingidos:

- Delimitação do sistema: definir seu escopo, seus principais requisitos, desejos do cliente e prazos;
- Definição da arquitetura candidata;
- Definição do protótipo das principais telas do sistema, baseadas nos principais casos de uso.

Esses objetivos são cumpridos através da geração dos seguintes artefatos:

- Documento de visão (Anexo A);
- Documento de arquitetura do sistema (Anexo B);
- Protótipos das principais telas do sistema.

Para (SCOTT, 2001) a arquitetura é definida como sendo uma organização fundamental do sistema, incluindo elementos estáticos, elementos dinâmicos, o modo como estes elementos trabalham juntos e o estilo arquitetônico total que guia a organização do sistema, proporcionando uma visão comum a todos os membros da equipe de desenvolvimento, a fim de tornar o sistema resultante adequadamente robusto, flexível, expansível e de custo viável. Nesse contexto, a arquitetura candidata nada mais é do que uma arquitetura que ainda não é a definitiva, ou seja, uma arquitetura que precisa ser validada.

DISCIPLINA DE REQUISITOS

A disciplina de requisitos na fase de concepção é aquela onde se gastará maior tempo e esforço. Ela inicia-se com a definição do projeto e termina com a breve descrição dos casos de uso. A figura 3.1 representa o seu fluxo de atividades.

A definição do projeto ocorre através de uma entrevista com o cliente, onde o mesmo expõe o problema que deseja solucionar. Se o problema for entendido pela equipe de desenvolvimento, passa-se para a definição do escopo, senão entrevista-se novamente o cliente, até que se tenha entendimento do problema.

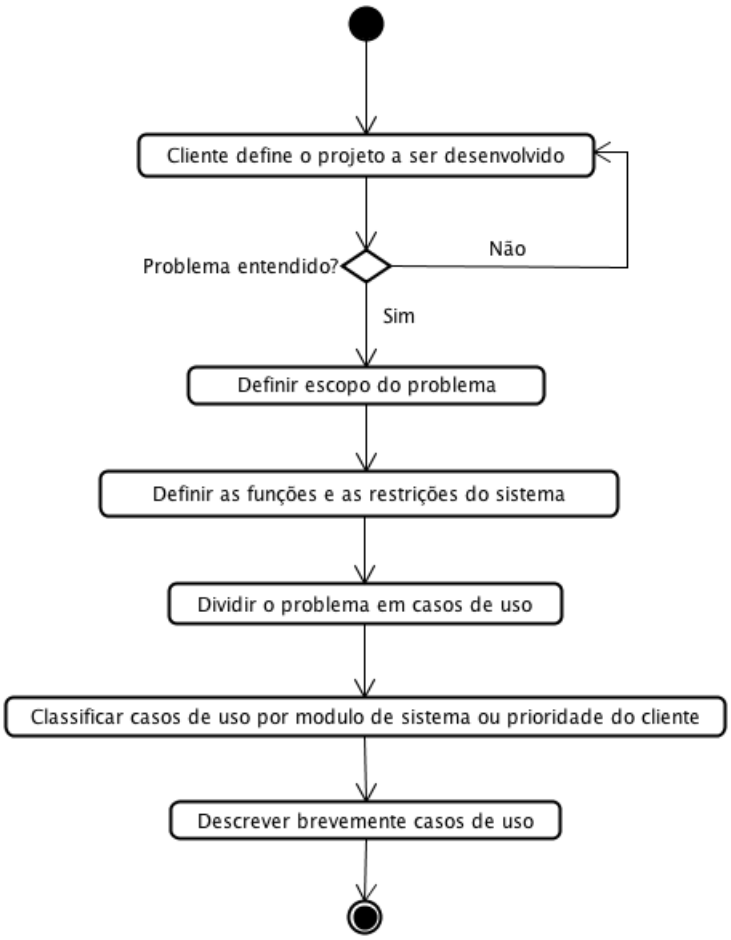

Figura 3.1: Fluxo de Atividades da Disciplina de Requisitos na fase de Concepção.
Fonte: Elaborada pelo autor.

Com o problema entendido, o seu escopo é definido. Esse escopo define o trabalho necessário para concluir o projeto, servindo como um guia (ou ponto de referência) para determinar que trabalho não está incluído (ou não é necessário) no projeto. Ou seja, após sua definição, sabe-se o que está dentro e o que está fora do projeto, de maneira clara e sem ambigüidades

Tendo definido o escopo, passa-se a definir as funções (requisitos funcionais) e as restrições (requisitos não-funcionais). De posse de tais requisitos, divide-se o problema em casos de uso. Esta atividade co-relaciona-se com a atividade de análise e projeto onde é construído o modelo de casos de uso. É importante lembrar que o processo é iterativo e as disciplinas ocorrem em paralelo dentro de uma fase.

Por fim, classifica-se os casos de uso por módulos do sistema ou por prioridade do cliente, e ao final, descreve-se brevemente os casos de uso e os atores participantes do sistema.

DISCIPLINA DE ANÁLISE E PROJETO

Na disciplina de Requisitos, o cliente foi ouvido, seus desejos com relação ao software foram transformados em requisitos e esses em casos de uso, conhecendo-se, assim, as possíveis funcionalidades do sistema. Entra em cena então a disciplina de análise e projeto. Grande parte do esforço da fase de concepção para definir uma arquitetura candidata concentra-se nesta disciplina. Ela inicia-se com a análise do ambiente físico e computacional e das restrições citadas pelo cliente, e é finalizada com a definição da arquitetura candidata. A figura 3.2 representa o seu fluxo de atividades.

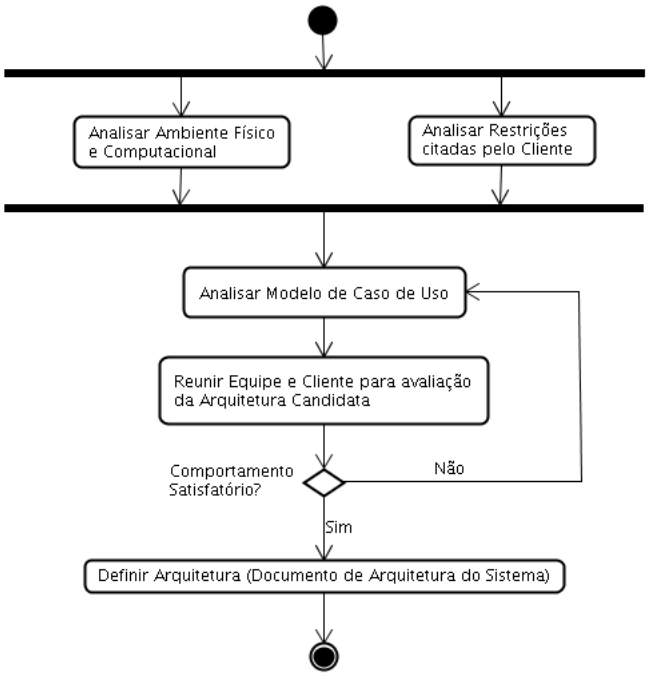

Figura 3.2: Fluxo de Atividades da Disciplina de Análise e Projeto na fase de Concepção.
Fonte: Elaborada pelo autor.

As atividades de análise, tanto do ambiente físico e computacional, como das restrições citadas pelo cliente, são executadas em paralelo. Elas servem para avaliar sobre que escopo deve ser pensada a arquitetura do sistema. Para que tais atividades ocorram, são necessários os insumos gerados pela disciplina de requisitos, ou seja, as definições captadas na entrevista com cliente (escopo e requisitos).

Concluídas as atividades acima, é feita uma análise do modelo de casos de uso. Tal atividade tem por objetivo verificar alguma possível incompatibilidade entre o modelo de casos de uso e as necessidade do cliente. Caso alguma incompatibilidade seja encontrada, ajusta-se o modelo de casos de uso.

Com o modelo de casos de uso refletindo as reais necessidade do cliente, é feita uma reunião

entre equipe de desenvolvimento e cliente, com a finalidade de avaliar a Arquitetura Candidata. Caso cliente e equipe estejam de acordo, elabora-se o documento com a arquitetura candidata. Caso contrário, é realizada uma nova análise do modelo de casos de uso, ajustando-se o que for necessário, sendo apresentado a arquitetura em uma nova reunião. Tais atividades se repetem até que cliente e equipe de desenvolvimento entrem em acordo.

DISCIPLINA DE IMPLEMENTAÇÃO E TESTE

O objetivo da implementação e teste na fase de concepção é a confecção do protótipo das principais telas do sistema. A figura 3.3 representa o seu fluxo de atividades.

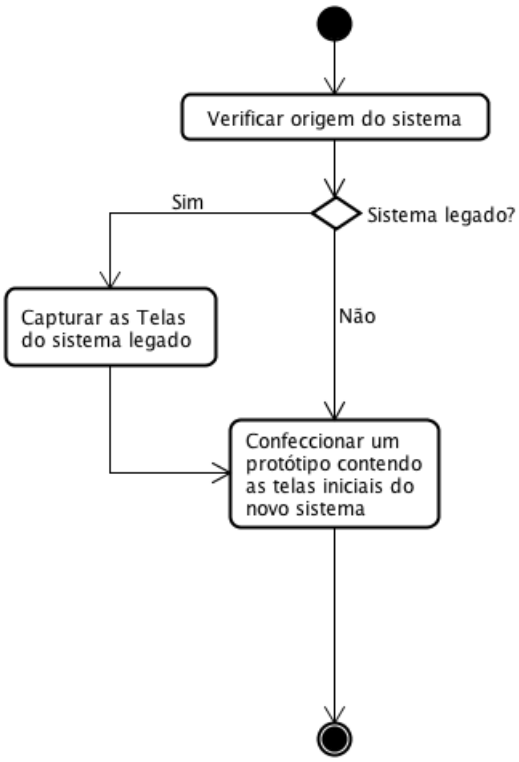

Figura 3.3: Fluxo de Atividades da Disciplina de Implementação e Teste na fase de Concepção.
Fonte: Elaborada pelo autor.

Como pode-se ver, seu fluxo é bastante simples. Caso o sistema a ser desenvolvido derive de um sistema legado, as telas do mesmo são capturadas e adaptadas ao novo sistema. Caso contrário, é confeccionado um protótipo contendo as principais telas do novo sistema.

3.1.2 FASE DE ELABORAÇÃO

O propósito da Fase de Elaboração é analisar o domínio do problema, validar uma arquitetura consistente, onde o sistema será desenvolvido, e desenvolver o plano de projeto. Embora

o processo sempre tenha que acomodar mudanças, as atividades desta fase asseguram que a arquitetura, requisitos e planos são bastante estáveis, podendo-se então, determinar o custo de maneira previsível e programar a conclusão do desenvolvimento.

A arquitetura candidata é transformada em uma base arquitetônica sólida através do desenvolvimento de um caso de uso que contemple ela por inteiro. Se ao final do desenvolvimento desse caso de uso, não for possível validar a arquitetura, deverá ser realizado o ajuste necessário, e deve-se iniciar uma nova iteração dentro da fase. Haverá tantas iterações quantas forem necessárias para que a arquitetura seja validada.

É importante ressaltar que o esforço para validar a arquitetura deve ser de toda a equipe, o que dará uma educação pela homogeneidade no desenvolvimento, tornando a Fase de Construção um notório complemento à esta fase. A sistemática de desenvolvimento das demais partes do sistema seguirá a que foi utilizada durante esta fase.

Ao final da fase, os seguintes objetivos devem ser atingidos:

- Definir, validar e delinear a arquitetura do sistema;
- Demonstrar que a arquitetura suportará os requisitos do sistema a um custo justo e em tempo justo;
- Estabelecer um ambiente de desenvolvimento.

Esses objetivos são cumpridos através da geração dos seguintes artefatos:

- Documento de visão revisado;
- Documento de detalhamento de caso de uso (Anexo C);
- Documento de arquitetura do sistema revisado;
- Plano de testes (Anexo D);
- Documento de configuração do ambiente (Anexo E).

DISCIPLINA DE REQUISITOS

O foco da Disciplina de Requisitos na Fase de Elaboração é o detalhamento inicial dos prováveis casos de uso que poderão ser utilizados para validar a arquitetura. A figura 3.4 mostra suas atividades.

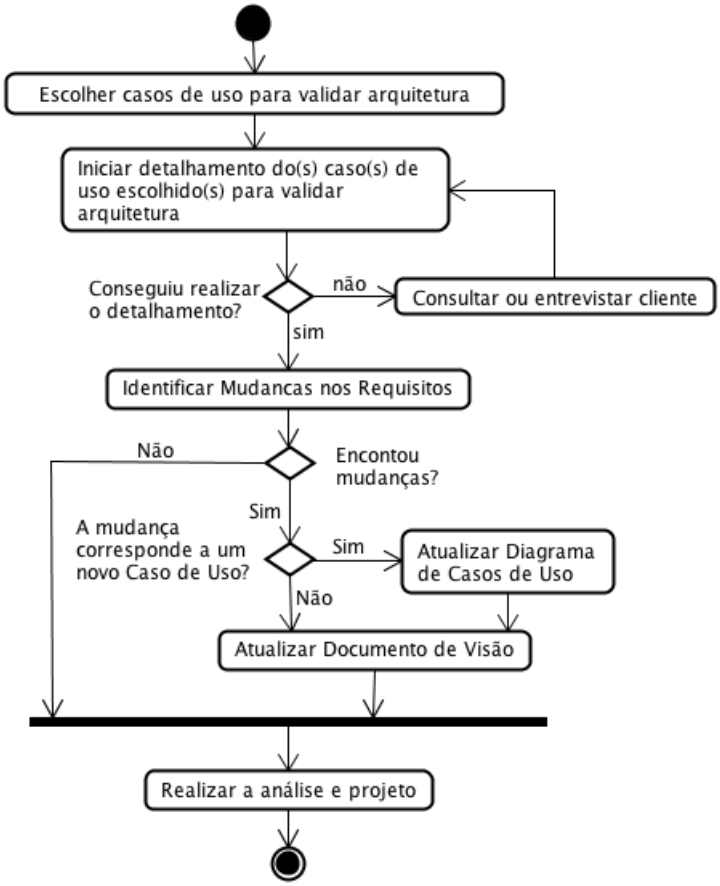

Figura 3.4: Fluxo de Atividades da Disciplina de Requisitos na fase de Elaboração.
Fonte: Elaborada pelo autor.

Antes de inciar o detalhamento, são escolhidos os casos de uso que poderão ser desenvolvidos para validar a arquiterura. Tal escolha é feita levando-se em consideração a complexidade e a abragência do mesmo em relação a arquitetura. Mais de uma caso de uso é escolhido pois, o detalhamento dos mesmos é quem vai definir qual deles é o mais adequado para realizar a validação da arquitetura.

Após escolhidos os casos de uso, é realizado o detalhamento inicial dos mesmos. Será confeccionado o Documento de Detalhamento de Caso de Uso, que conterá as informações necessárias para que a próxima disciplina, análise e projeto, possa ser realizada com sucesso.

Caso venham a surgir dúvidas, inseguranças ou inconsistências no detalhamento, o cliente será entrevistado ou consultado por algum meio de comunicação. Após sanar a dúvida, realiza-se novamente a primeira atividade, ajustando o que for necessário.

Após detalhar o caso de uso, é feita uma revisão nos requisitos, o que pode ocasionar o surgimento de um novo caso de uso, implicando em uma revisão no diagrama de casos de uso e também no documento de visão. Concluídas todas as atividades, inicia-se a disciplina de

Análise e Projeto.

DISCIPLINA DE ANÁLISE E PROJETO

A disciplina de Análise e Projeto desempenha um papel importante para que o objetivo da fase de Elaboração seja atingido. Ela é quem dá forma ao domínio do sistema, gerando um modelo conceitual orientado pelos casos de uso. Modelos estáticos e dinâmicos são formulados e servirão como base para a implementação. A figura 3.4 representa o fluxo de atividades da disciplina.

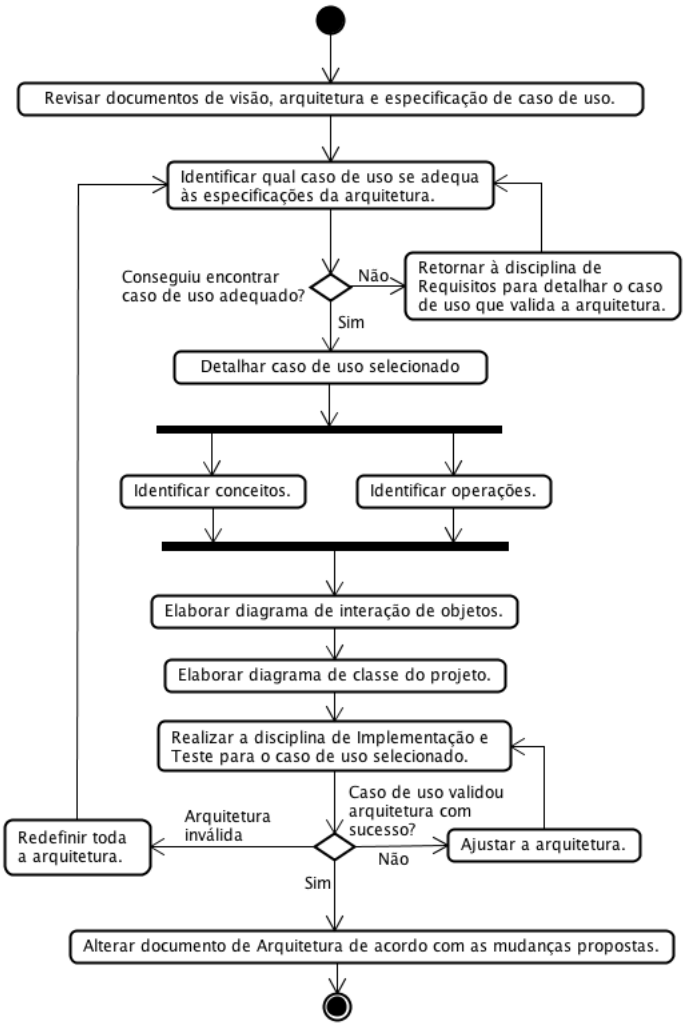

Figura 3.5: Fluxo de Atividades da Disciplina de Análise e Projeto na fase de Elaboração.
Fonte: Elaborada pelo autor.

A primeira atividade a ser realizada nesta disciplina é a revisão do documento de visão, documento de arquitetura e documentos de detalhamento de caso de uso, possibilitando assim, uma escolha mais adequada do caso de uso a ser utilizado na validação da arquitetura. Em seguida é identificado o caso de uso que melhor se adeque as especificações da arquitetura.

Se nenhum dos casos de uso detalhados na disciplina de Requisitos atender as especificações da arquitetura, deve-se realizar novamente a disciplina de requisitos, detalhando assim outros casos de uso que possam atender a tais especificações.

Quando o caso de uso for identificado, o mesmo será analisado, buscando identificar conceitos e operações. Em seguida, elabora-se o diagrama de iteração de objetos (Diagrama de Sequência) e o diagrama de classe do projeto. Por fim, a disciplina de implementação e teste é inciada.

Se ao final da implementação e teste, a arquitetura não for validada, a mesma deve ser ajustada e a disciplina de implementação e teste executada novamente. Caso seja totalmente inválida, a arquitetura deve ser redefinida, e a disciplina análise e projeto deve ser ativada novamente. A arquitetura estando válida, o documento de Arquitetura deve ser atualizado, caso exista alguma mudança.

DISCIPLINA DE IMPLEMENTAÇÃO E TESTE

Nas disciplinas anteriores, o caso de uso que justifica toda a Arquitetura foi detalhado, analisado e projetado. Cabe agora a implementação e teste do mesmo. Ao final dessa disciplina, o caso de uso deverá ter sido construído seguindo a arquitetura planejada no documento de arquitetura. A figura 3.4 representa o fluxo de atividades da disciplina.

Inicialmente é realizado um estudo no documento de detalhamento do caso de uso que será implementado, construído na disciplina de Análise e Projeto.De posse de todas as informações necessárias, os testes unitário e funcional são construídos, as classes do componente são implementadas e os testes efetuados. Caso erros sejam encontrados após os testes, o componente deve ter sua implementação revisada, para que os erros possam ser corrigidos.

As atividades dessa disciplina demonstram uma preocupação quanto aos testes, obrigando o aluno a planejar os testes antes de realizar a implementação, garantindo assim que tudo que for desenvolvido deverá ser testado.

Ao fim desta disciplina, a equipe pode gozar de uma liberdade tal que permita aos trabalhadores a avaliação de alternativas tecnológicas, podendo-se assim adotar novas tecnologias, agregando valor ao sistema.

3.1.3 FASE DE CONSTRUÇÃO

Durante a fase de construção, todos os componentes e características restantes da aplicação são desenvolvidos, testados e integrados ao produto. A fase de construção é, de certo modo,

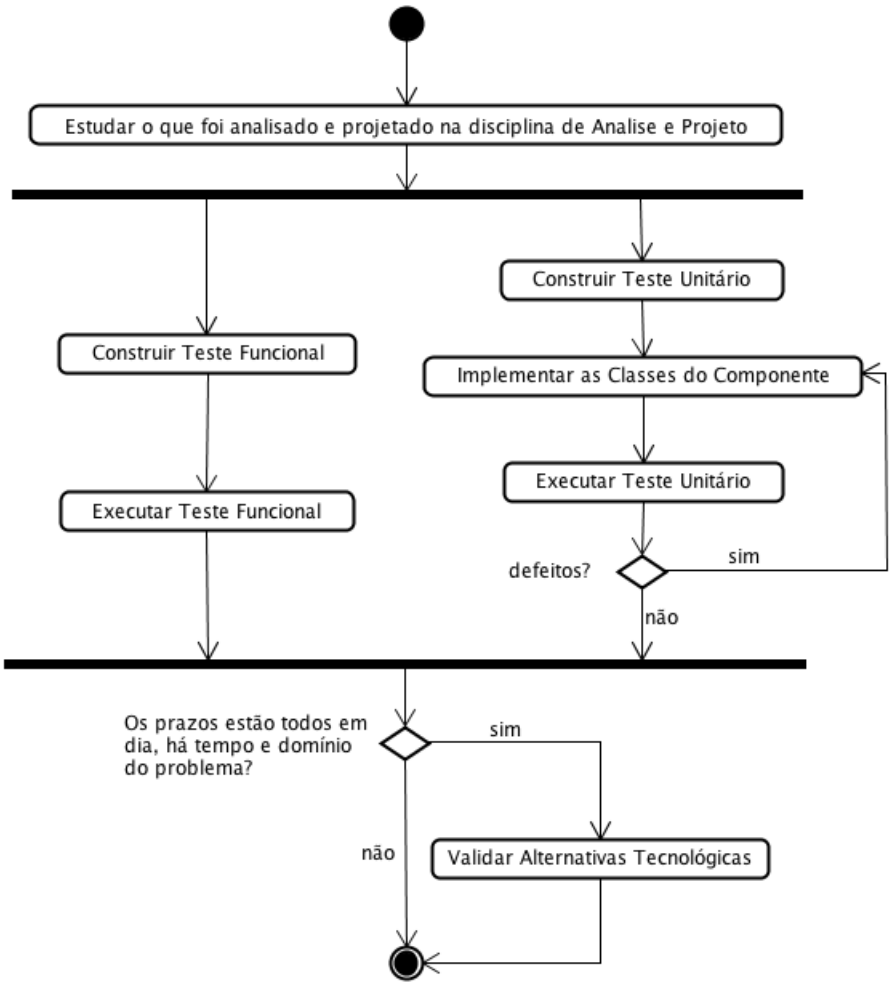

Figura 3.6: Fluxo de Atividades da Disciplina de Implementação e Teste na fase de Elaboração. Fonte: Elaborada pelo autor.

um processo industrial, no qual é enfocado o gerenciamento de recursos, prazos e qualidade. Certamente, o quebra-cabeça do gerenciamento sofre uma transição do desenvolvimento de propriedade intelectual durante a concepção e a elaboração para o desenvolvimento de produtos para distribuição durante a construção e a validação. A meta da fase de construção é esclarecer os requisitos restantes e concluir o desenvolvimento do sistema com base na arquitetura.

Nessa fase, a equipe poderá ser dividida, possibilitando a execução das iterações de maneira paralela. Cada iteração corresponde ao desenvolvimento de um caso de uso. Portanto, nesta fase, o número de iterações corresponderá ao número de casos de uso a serem desenvolvidos.

Ao final da fase, os seguintes objetivos devem ser atingidos:

- Atingir a qualidade adequada com rapidez e eficiência;
- Concluir a análise, o desenvolvimento e o teste de todas as funcionalidades necessárias;

- Desenvolver de modo iterativo e incremental um produto completo que esteja pronto para a validação;
- Minimizar custos de desenvolvimento, aperfeiçoando recursos e evitando fragmentar e refazer o desnecessário.

Esses objetivos são cumpridos através da geração dos seguintes artefatos:

- Documento de visão revisado;
- Documento de detalhamento de caso de uso, para cada caso de uso implementado;
- Código dos casos de uso implementados;
- Plano de integração (Anexo F).

DISCIPLINA DE REQUISITOS

As atividades da disciplina de Requisitos nesta fase são praticamente as mesmas da fase anterior. Elas diferem apenas na quantidade de casos de uso detalhados. Na fase anterior são detalhados os prováveis casos de uso que podem validar a arquitetura. Já nesta fase, apenas o caso de uso participante da iteração é detalhado. O seu fluxo de atividades é mostrado na figura 3.7.

DISCIPLINA DE ANÁLISE E PROJETO

Os objetivos da disciplina de Análise e Projeto nesta fase são os mesmos da fase anterior. Como não existe a preocupação em validar a arquitetura, o fluxo de atividades, figura 3.8, é uma simplificação do anterior.

As atividades consitem basicamente em analisar o caso de uso, identificando conceitos, operações e componentes; atividades que podem acontecer em paralelo. Em seguida o diagrama de iteração de objetos é construído e o diagrama de classes do projeto atualizado. É importante salientar que é nessa disciplina que os componentes são identificados. Eles serão a base para a implementação, que ocorrerá orientada a componentes.

DISCIPLINA DE IMPLEMENTAÇÃO E TESTE

Com a arquitetura do sistema validada na fase de Elaboração, esta disciplina agora tem como objetivo realizar a codificação, ou seja, a implementação e o teste dos componentes per-

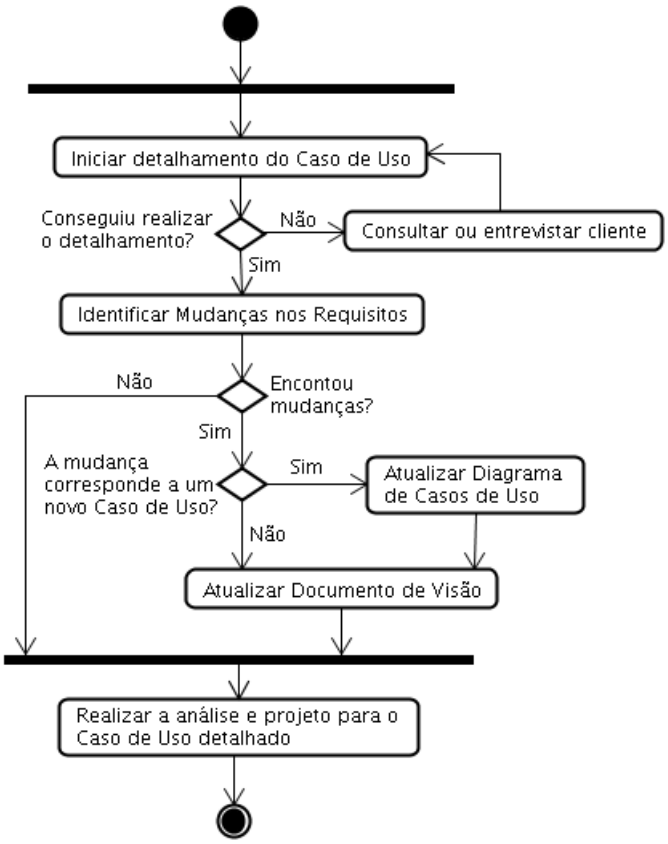

Figura 3.7: Fluxo de Atividades da Disciplina de Requisitos na fase de Construção.
Fonte: Elaborada pelo autor.

tencentes ao caso de uso, bem como sua integração com a parte do sistema já desenvolvida. O fluxo de atividas é representado pela figura 3.9.

A primeira atividade da disciplina é o planjemento da integração a ser realizada ao final da mesma. Em seguida, os componentes identificados na disciplina anterior, serão divididos dentro da equipe de desenvolvimento, de maneiras que a codificação dos mesmos ocorra de maneira paralela.

A codificação ocorre da mesma maneira que na fase anterior, sendo inicialmente construídos os testes unitários e em seguida, implementado e testado o componente. Caso erros sejam encontrados após os testes, o componente é implementado novamente, para que os erros possam ser corrigidos. A iteração acaba quando todos os componentes para o caso de uso forem integrados ao sistema. Em cada iteração dentro da fase, será implementado um caso de uso.

3.1.4 FASE DE VALIDAÇÃO

O foco da Fase de Validação é assegurar que o software esteja disponível para seus usuários finais. Ela pode atravessar várias iterações e inclui testes do produto em preparação para *rele-*

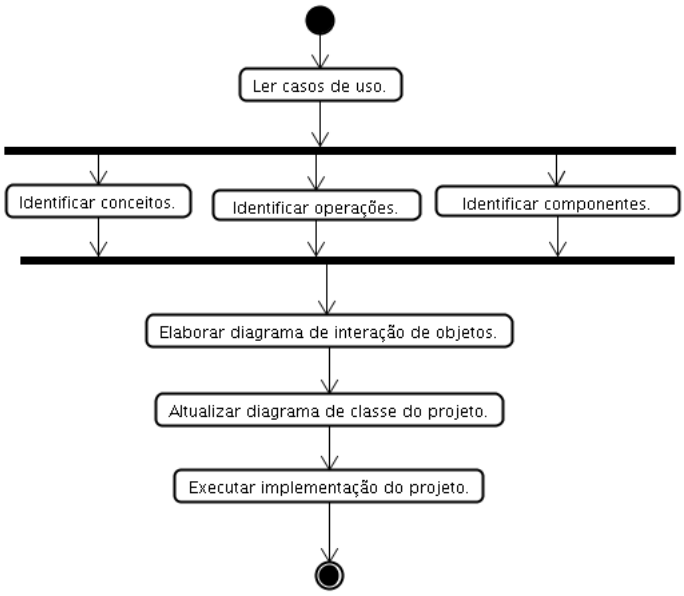

Figura 3.8: Fluxo de Atividades da Disciplina de Análise e Projeto na fase de Construção.
Fonte: Elaborada pelo autor.

ase e ajustes pequenos com base no *feedback* do usuário. Nesse momento do ciclo de vida, o *feedback* do usuário deve priorizar o ajuste fino do produto, a configuração, a instalação e os problemas de usabilidade; todos os problemas estruturais mais graves devem ter sido trabalhados antes no ciclo de vida do projeto.

No entanto, modificações podem acontecer quando o cliente for realizar o "teste" da aplicação. Essas modificações incluem desde pequenos ajustes na interface, até a inclusão de novas características, o que gera o surgimento de novos requisitos e conseqüentemente novos casos de uso. Caso ocorram mudanças a nível de requitos e casos de uso, será realizada uma nova iteração para cada novo caso de uso, com basicamente as mesmas atividades da fase de construção. Isso não significa que a fase de construção irá acontecer novamente, apenas as atividades executas é que serão as mesmas.

No fim do ciclo de vida da fase, os objetivos devem ter sido atendidos e o projeto deve estar em uma posição de fechamento. Em alguns casos, o fim do ciclo de vida atual pode coincidir com o início de outro ciclo de vida no mesmo produto, conduzindo à nova geração ou versão do produto. Para outros projetos, o fim da validação pode coincidir com uma liberação total dos artefatos a terceiros, que poderão ser responsáveis pela operação, manutenção e melhorias no sistema.

Ao final da fase, os seguintes objetivos devem ser atingidos:

- Entregar e instalar o produto gerado no cliente,

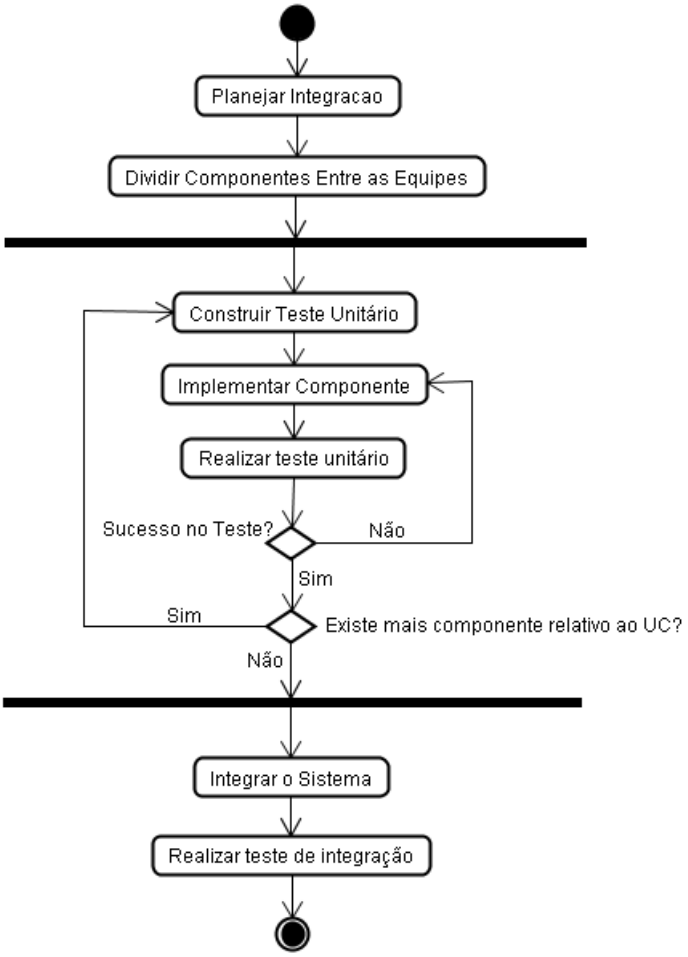

Figura 3.9: Fluxo de Atividades da Disciplina de Implementação e Teste na fase de Construção. Fonte: Elaborada pelo autor.

- Corrigir defeitos e modificar o sistema para corrigir problemas não identificados previamente.

Esses objetivos são cumpridos através da geração do Produto.

DISCIPLINA DE REQUISITOS

A disciplina de requisitos na fase de validação é bastante simples. Ela será executada apenas se novas funcionalidades tiverem sido adicionadas ao sistema. O seu fluxo, figura 3.10, possui apenas duas atividades. Uma responsável por verificar o surgimento de novas funcionalidades e, em caso positivo, outra para chamar as atividades da mesma disciplina da fase anterior.

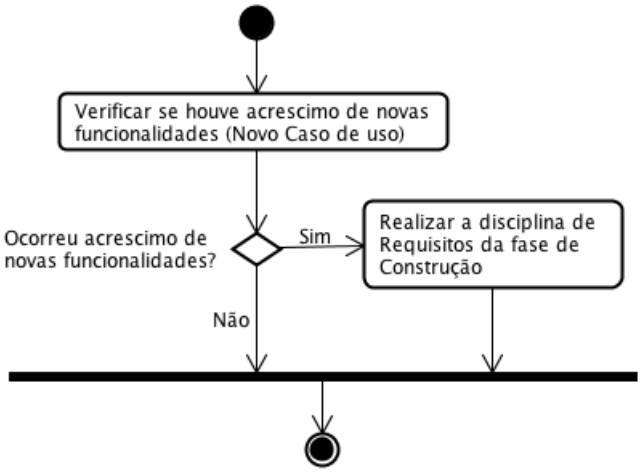

Figura 3.10: Fluxo de Atividades da Disciplina de Requisitos na fase de Validação.
Fonte: Elaborada pelo autor.

DISCIPLINA DE ANÁLISE E PROJETO

A disciplina de analise e projeto é tão simples quanto a anterior. O seu fluxo, figura 3.11, possui as mesmas atividades da anterior, com a única diferença de que, em caso de novas funcionalidades, serão realizadas as atividades da disciplina de análise e projeto da fase anterior.

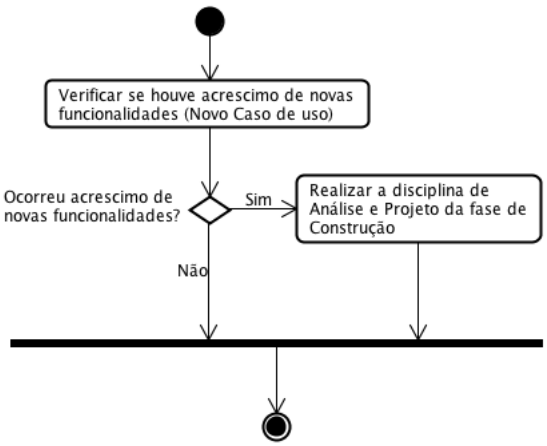

Figura 3.11: Fluxo de Atividades da Disciplina de Análise e Projeto na fase de Validação.
Fonte: Elaborada pelo autor.

DISCIPLINA DE IMPLEMENTAÇÃO E TESTE

Nesta fase, o foco principal da disciplina de Implementação é realizar os testes do produto em preparação para release, bem como ajustes pequenos com base no *feedback* do cliente. Trata-se de uma disciplina intensa e plenamente relevante nesta fase do projeto. Suas atividades estão representadas no fluxo da figura 3.12.

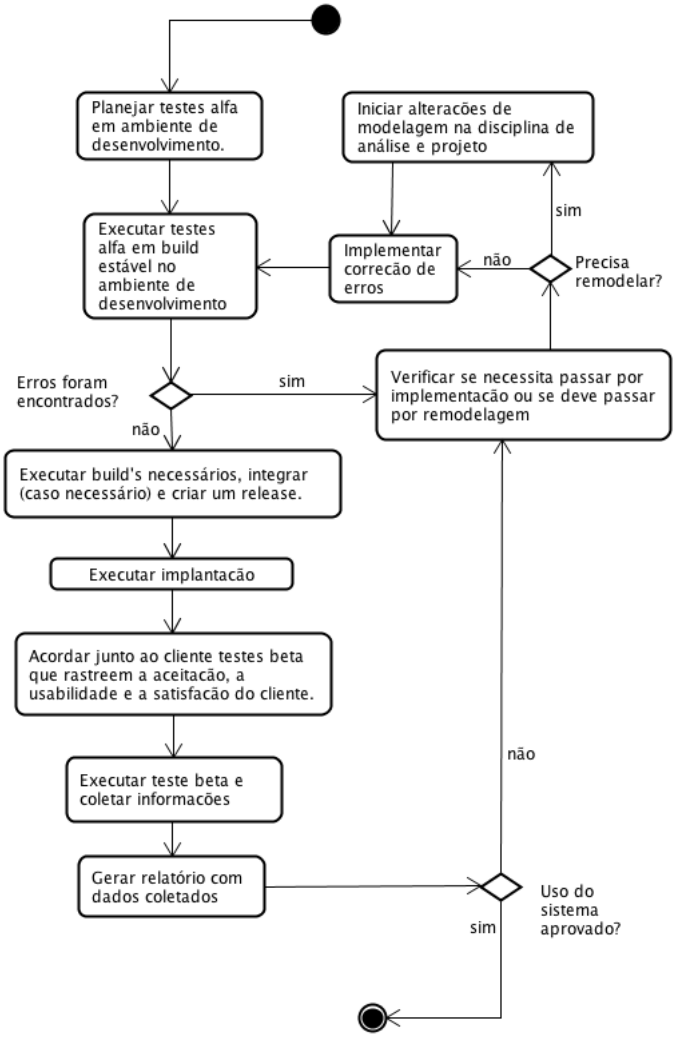

Figura 3.12: Fluxo de Atividades da Disciplina de Implementação e Teste na fase de Validação. Fonte: Elaborada pelo autor.

As atividades acontecem básicamente em dois momentos. No primeiro momento são planejados e executados testes no ambiente de desenvolvimento (testes alfa) e no segundo, o sistema é implantado e são executados testes de usabilidade e testes beta (teste no ambiente do cliente). Ao final, é gerado um relatório com as informações coletadas nos testes.

Encontrando erros após a execuçãoo dos testes alfa e/ou beta, dependo de sua natuteza, a correção deverá ser feita na modelagem ou no código fonte. No caso de uma remodelagem, as disciplinas de requisitos e análise e projeto são executadas e ao final, realiza-se novamente os referidos testes. Já no caso de problemas no código fonte, uma nova execução da disciplina de implementacao e testes deverá sanar o problema. São realizadas tantas iterações quantas forem necessárias, até que o uso do sistema seja aprovado pelo cliente.

4 SIGES: ESTUDO DE CASO APLICANDO O PAS

Este capítulo tem por objetivo descrever a aplicação do PAS no desenvolvimento do SIGES. Ele foi estruturado, inicialmente, com a delimitação do estudo de caso, levando-se em consideração o escopo do sistema e a equipe de desenvolvimento. Em seguida, será descrita a aplicação de cada fase do processo, indicando as atividades realizadas e os seus resultados. Todos os artefatos gerados durante a execução do processo encontram-se disponíveis na página do sistema[1], construída utilizando a ferramenta Trac, explicada mais adiante.

4.1 DELIMITAÇÃO DO ESTUDO DE CASO

4.1.1 O SISTEMA E SUA FRONTEIRA

O SIGES é um sistema de gerenciamento das escalas de serviço destinado às Organizações Militares do Exército Brasileiro. Escala de serviço é a relação do pessoal ou das frações de tropa que concorrem na execução de determinado serviço, tendo por finalidade principal a distribuição eqüitativa de todos os serviços de uma OM.

Para um serviço ser considerado de escala, ele não deve ser atribuído permanentemente à mesma pessoa, ou fração de tropa, não importando em delegação pessoal ou escolha, devendo obedecer à algumas regras[2]. O sistema terá seu funcionamento baseado nessas regras e deverá permitir o cadastro dos serviços executados em uma OM e dos integrantes do mesmo, formando assim a escala de serviço. A escala poderá ser dividida em vermelha (dias sem expediente) e preta (dias com expediente) bem como poderá ser initerrupta/ocasional e externa/interna.

De posse dessas informações, o sistema deverá controlar as folgas e determinar quem estará de serviço no dia seguinte, bem como determinar a previsão da escala para um período de 1 (um) mês.

4.1.2 A EQUIPE DE DESENVOLVIMENTO

Apesar de não existir a especificação de papéis no PAS, foi realizada uma divisão da equipe em papéis, visando uma melhor distribuição do trabalho e aproveitamento das expetirses dos desenvolvedores. A equipe foi composta por sete pessoas, sendo um gerente e os demais desen-

[1] http://trac.itapirunet.com.br/wiki/siges

[2] Disponível em http://trac.itapirunet.com.br/wiki/siges/documentacao/regras

volvedores. A principal atribuição do gerente foi garantir que o processo fosse seguido e que, todos os artefatos fossem gerados. Coube a ele ainda a estruturação da página a ser usada pela equipe, bem como a ligação com o cliente do sistema, sanando qualquer dúvida que surgisse durante o desenvolvimento.

O restante dos membros da equipe foram divididos em 5 (cinco) papéis, sendo cada um responsável por aquilo que lhe foi designado de acordo com suas habilidades, podendo, no entanto, interferir em um papel que não fosse o seu, desde que solicitado. Alguns membros acumularam mais de um papel, dada a proximidade das funções. A tabela 4.1 descreve as atividades desenvolvidas por cada papel.

Tabela 4.1: Papéis da equipe

Papel	**Descrição**
Engenheiro de Software	Responsável pela criação, manutenção e auditoria de metodologias de desenvolvimento de sistemas. Realiza também o acompanhamento das métricas de desempenho e qualidade dos produtos gerados, comparando-as com as métricas-padrão do mercado e adequando quando for necessário.
Analista de Sistema	Realiza básicamente a análise de requisitos e o projeto a ser seguido pelo desenvolvedor. A disciplina de análise e projeto é de sua responsabilidade.
Desenvolvedor	É o implementador do software. Responsável por desenvolver o software de acordo com as restrições do cliente, tanto com relação as tecnologias utilizadas quanto com ao ambiente físico e tempo, seguindo o que foi projetado pelo análista.
Analista de interface gráfica e interação com o cliente	Efetua atividades relacionadas ao design do sistema, bem como efetua testes de usabilidade junto ao cliente.
Testador	É o responsável por construir os testes e efetuá-los em cima do que foi implementado, preocupando-se com o desempenho e a funcionalidade.

Fonte: Elaborada pelo autor.

4.1.3 O AMBIENTE DE TRABALHO

Visando um melhor aproveitamento por parte da equipe de desenvolvimento, foram utilizadas as seguintes ferramentas: Subversion e Trac. O Subversion é um sistema de controle de versão livre/open-source que gerencia arquivos e diretórios ao longo do tempo. O Trac é uma ferramenta minimalista, open source e de interface web que ajuda o desenvolvedor a rastrear mudanças, entender o porquê de cada uma e medir o seu impacto no projeto como um todo. Ele funciona integrado ao SubVersion.

Apesar da facilidade proporcionada pelas ferramentas citadas anteriormente, foi necessário

a realização de um treinamento, com o objetivo de educar a equipe no sentido de utilizar as tecnologias de maneira adequada. Foi realizado então, um treinamento do fluxo de trabalho (Figura 4.1) a ser utilizado ao longo do projeto, simulando o uso do SVN na criação de um mini-projeto.

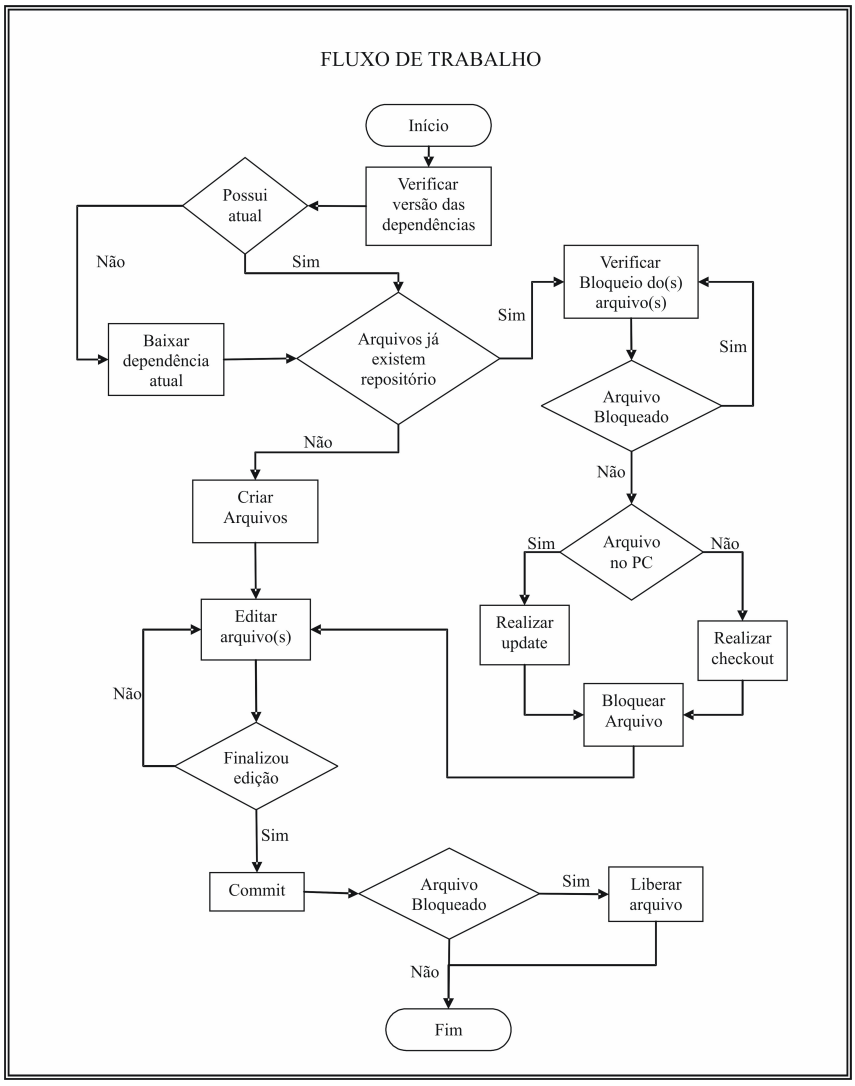

Figura 4.1: Fluxo de trabalho para controle de versão.
Fonte: Elaborada pelo autor.

A ferramenta utilizada para o desenvolvimento de código fonte foi o Eclipse[3]. A escolha deu-se por se tratar de uma IDE bastante poderosa e customizável, pois utiliza o conceito de *plugins*, para que novas funcionalidades sejam adicionadas. Tanto o Sistema de Gerenciamento de Bancos de Dados (SGBD), como o Servidor de Aplicação, não foram escolhidos pela equipe de desenvolvimento, mas impostos pelo cliente, já que a plataforma onde o sistema iria funcionar já existia e não podia ser alterada.

[3]Disponível em http://www.eclipse.org

A tabela 4.2 descreve todos os softwares utilizados, a sua versão e a maneira como o mesmo foi utilizado.

Tabela 4.2: Softwares utilizados

Software	**Versão**	**Uso**
Trac	0.10	Desenvolvimento colaborativo e rastreamento de mudanças.
SVN	1.4.0	Controle de Versão.
Eclipse WTP	3.2	Desenvolvimento de códigos fontes para JSF, JSTL, EJB3, JPA.
PostgreSQL	8.2.3-1	SGBD.
JBoss JEMS	1.2.0	Servidor da aplicação.

Fonte: Elaborada pelo autor.

Com relação aos recursos de hardware, existia a disposição da equipe um servidor linux instalado no cliente, um servidor linux e 02 (duas) máquinas de desenvolvimento Intel Pentium 4 2.8 GHz com 512MB de memória RAM, ambos instalados no laboratório de PDS-I e 3 (três) máquinas dos próprios integrantes da equipe, com Core Duo 1.66 GHz e 512MB de memória RAM.

4.2 EXECUÇÃO DAS FASES

4.2.1 FASE DE CONCEPÇÃO

A fase de concepção teve início no dia 20/03/2007 e foi concluída em 10/04/2007, tendo sido executada em uma única iteração. Todos os objetivos propostos para essa fase foram alcançados, bem como todos os artefatos previstos gerados. A equipe de desenvolvimento não encontrou nenhum obstáculo durante a execução da fase, conseguindo entender e executar todos os fluxogramas de atividades das disciplinas.

O principal artefato gerado nessa fase foi o documento de visão[4], onde foram definidos escopo, principais requisitos e casos de uso do sistema. A figura 4.2 ilustra o diagrama de casos de uso, principal elemento do documento de visão.

A tabelas 4.3, 4.4 e 4.5 descrevem, respectivamente, as atividades executadas nas disciplinas de Requisitos, Análise e Projeto e Implementação e Teste. Foram omitidas as atividades que não necessitaram ser realizadas.

[4]Pode ser obtido em http://trac.itapirunet.com.br/attachment/wiki/siges/processo/concepcao/siges_doc_visao.pdf

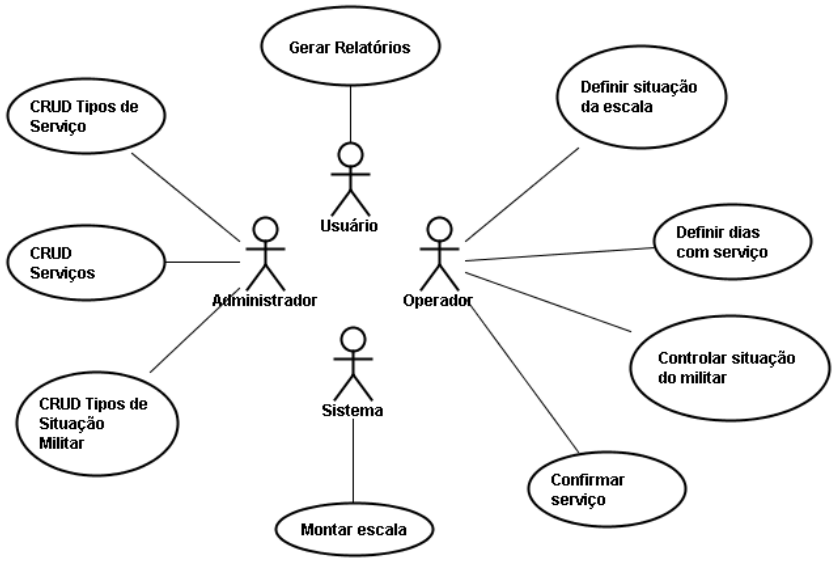

Figura 4.2: Diagrama de Casos de Uso.
Fonte: Elaborada pelo autor.

4.2.2 FASE DE ELABORAÇÃO

A fase de elaboração teve início no dia 10/04/2007 e foi concluída em 01/06/2007, com a validação do cliente em relação a interface grática e a arquitetura do sistema, sendo executada em apenas uma iteração. Todos os objetivos propostos para essa fase foram alcançados, bem como todos os artefatos previstos gerados. Por várias vezes a equipe necessitou da presença do cliente para sanar dúvidas relacionadas as regras de negócio do sistema, sendo, na maioria da vezes, prontamente atendidada. A atenção e a proximidade do cliente foram vitais para o sucesso desta fase. Da mesma maneira que na fase anterior, a equipe não encontrou nenhum obstáculo durante a execução da fase, conseguindo entender e executar todos os fluxogramas de atividades das disciplinas.

O principal objetivo dessa fase foi a validação da arquitetura candidata, proposta na fase anterior. Esse objetivo foi alcançado sem muitas dificuldades, fruto de uma escolha correta e de um detalhamento bem feito no caso de uso CRUD[5] Serviço[6]. Com a arquitetura válida, o documento de arquitetura do sistema[7] foi enfim, concluído.

A tabelas 4.6, 4.7 e 4.8 descrevem, respectivamente, as atividades executadas nas disciplinas de Requisitos, Análise e Projeto e Implementação e Teste. Foram omitidas as atividades que não necessitaram ser realizadas.

[5]CRUD é o acrônimo de ***C**reate, **R**ead, **U**pdate, **D**elete.*

[6]Documento de Detalhamento disponível em http://trac.itapirunet.com.br/browser/SIGES/Documentacao/Detalhamento de casos de uso/caso de uso CRUD servicos/Descricao_Caso_de_uso.pdf

[7]Pode ser obtido em http://trac.itapirunet.com.br/browser/SIGES/Documentacao/Arquitetura do sistema/Modelo de arquitetura.doc

Tabela 4.3: Atividades da Disciplina de Requisitos - Concepção

Data	**Atividade**	**Execução**
20/03/2007	Definição de qual será o projeto pelo cliente.	Junto a representação do cliente, foi definido que seria desenvolvido o Sistema de gerenciamento de escala de serviço (SIGES).
26/03/2007	Definir escopo do Sistema.	Foi definido pelo cliente e pela equipe de desenvolvimento o escopo do sistema.
30/03/2007	Definir funções e Restrições do sistema.	Foi desenvolvido um documento de visão, no qual as funções e restrições do sistema são definidos.
02/04/2007	Dividir o problema em casos de uso.	A partir das funções e restrições do sistema, o mesmo foi dividido em casos de uso.
07/04/2007	Classificar casos de uso por modulo do sistema ou por prioridade do cliente.	A classificação foi definida através do critério de prioridade do cliente.
09/04/2007	Descrever brevemente casos de uso.	A partir do documento de visão já construído foi feita uma breve descrição dos casos de uso já identificados anteriormente.

Fonte: Elaborada pelo autor.

4.2.3 FASE DE CONSTRUÇÃO

A fase de construção foi iniciada no dia 01/06/2007 e concluída em 31/08/2007, após a execução da sexta iteração. A execução das seis iterações não permitiu a construção do sistema como um todo, tendo sido desenvolvidos apenas os seguintes casos de uso:

- CRUD Patente
- CRUD Situação Militar
- Montar Escala
- Alterar Situação Militar
- Alterar Situação escala
- Definir Dias com Serviço

Por se tratar da fase onde o objetivo principal é a construção de um produto completo, surgiram muitas dúvidas relacionadas as regras de negócio, tornando necessário várias reuniões com o cliente, conseguindo-se sanar todas as dúvidas a contento. Apesar das dificuldades, conseguiu-se construir um produto funcional, que não foi concluído devido ao fator tempo, e não por deficiência no processo de software.

Tabela 4.4: Atividades da Disciplina de Análise e Projeto - Concepção

Data	**Atividade**	**Execução**
09/04/2007	Analisar ambiente físico e computacional.	A partir de informações passadas pelo cliente foram identificados os ambientes físicos e computacionais para o desenvolvimento da aplicação. Além de contar com o servidor do cliente, foi passado a disposição da equipe o servidor de PDS-I e máquinas para o desenvolvimento.
09/04/2007	Analisar restrições citadas pelo cliente.	As restrições já identificadas na disciplina anterior foram analisadas.
09/04/2007	Analisar modelo de casos de uso.	A partir de informações do artefato gerado na disciplina de requisitos, foram analisados os casos de uso.
10/04/2007	Reunir equipe e cliente para a avaliação da arquitetura candidata.	A arquitetura candidata foi definida e apresentada ao cliente.
10/04/2007	Definir arquitetura (Documento de arquitetura do sistema)	Estando o cliente e a equipe de desenvolvimento de acordo, o documento de arquitetura foi criado, com base na arquitetura candidata.

Fonte: Elaborada pelo autor.

Tabela 4.5: Atividades da Disciplina de Implementação e Teste - Concepção

Data	**Atividade**	**Execução**
10/04/2007	Confeccionar um prótotipo contendo as telas iniciais do novo sistema.	Os prótotipos das interfaces do sistema foram criadas e apresentadas ao cliente, sendo aprovadas.

Fonte: Elaborada pelo autor.

Houve por parte da equipe de desenvolvimento um entendimento dos fluxogramas de atividades das disciplinas desenvolvidas ao longo da fase, já que foram executadas em sua completude nas seis iterações realizadas. No entanto, a execução de testes, tanto unitário, quanto de integração, não foram realizados a contento, ficando em segundo plano, ou até mesmo, sem ser executados.

A tabelas 4.9, 4.10 e 4.11 descrevem, respectivamente, as atividades executadas nas disciplinas de Requisitos, Análise e Projeto e Implementação e Teste, referentes ao desenvolvimento do caso de uso mais complexo do sistema, o Montar Escala. O desenvolvimento dos demais casos de uso ocorreu de maneira semelhante, seguindo a mesma linha de raciocínio. As atividades que não foram realizadas estão omitidas.

Tabela 4.6: Atividades da Disciplina de Requisitos - Elaboração

Data	**Atividade**	**Execução**
10/04/07	Iniciar detalhamento do(s) caso(s) de uso escolhido(s) para validar arquitetura e riscos.	Apenas um caso de uso, CRUD Serviço, foi escolhido para ser utilizado na validação da arquitetura. O detalhamento foi realizado pelo engenheiro de software da equipe, tendo como base a arquitetura candidata definida na fase anterior.
10/04/07	Consultar ou entrevistar cliente	Por diversas vezes foi necessário consultar o cliente para esclarecimento sobre o funcionamento do caso de uso CRUD Serviço.
10/04/07	Identificar mudanças nos requisitos	Os requisitos sofreram poucas alterações, resumindo-se a mudanças na descrição dos mesmos. O cronograma, bastante alterado, foi devidamente atualizado.
10/04/07	Atualizar Documento de Visão	Como não surgiu nenhum caso de uso novo, o documento de visão foi revisado apenas com as alterações nos requisitos.

Fonte: Elaborada pelo autor.

Tabela 4.7: Atividades da Disciplina de Análise e Projeto - Elaboração

Data	**Atividade**	**Execução**
10/04/07	Revisar documento de visão, arquitetura e especificação de caso de uso	A equipe reuniu-se para analisar os documentos e ficar em condições de realizar o restante do detalhamento do caso de uso CRUD Serviço.
10/04/07	Identificar qual caso de uso se adequa às especificações da arquitetura	Não foi necessário realizar essa atividade, já que apenas um caso de uso foi escolhido como sendo adequado às especificações da arquitetura.
16/04/07	Ler caso de uso	A equipe reuniu-se para esclarecer e listar todas as dúvidas em relaçao ao caso de uso
16/04/07	Identificar conceitos	Através de uma reunião com o cliente, foram esclarecidos todos os conceitos em relação ao caso de uso CRUD Serviço
16/04/07	Identificar operações	Através de uma reunião com o cliente, foram esclarecidas todas as operações do caso de uso CRUD Serviço
27/04/07	Elaborar diagrama de interação de objetos	O diagrama de interação de objetos foi construído pelo engenheiro da equipe
27/04/07	Elaborar diagrama de classe do projeto	O diagrama de classe foi construído pelo engenheiro da equipe
27/04/07	Implementar caso de uso	Foi iniciada a disciplina de implementação e teste. O caso de uso foi implementado em um período de 4 semanas
01/06/07	Alterar documento de Arquitetura de acordo com as mudanças propostas	Através de uma reunião com o cliente, o caso de uso foi validado, não sendo necessário alterar nada no documento de arquitetura.

Fonte: Elaborada pelo autor.

Tabela 4.8: Atividades da Disciplina de Implementação e Teste - Elaboração

Data	**Atividade**	**Execução**
16/04/07	Ler documento de arquitetura do sistema, identificar camadas e componentes	Para este caso de uso foram identificados 3 componentes, distribuídos nas camadas domínio, persistência e serviço.
27/04/07	Estudar o que foi analisado e projetado na disciplina de Análise e Projeto	Todos os membros da equipe estudaram o documento de detalhamento do caso de uso.
27/04/07	Planejar teste funcional e unitário	Documento de planejamento de testes foi elaborado pelo engenheiro da equipe.
27/04/07	Implementar as classes do componente	As classes necessárias foram implementadas dentro de suas camadas, ou seja, de acordo com a arquitetura proposta.
27/04/07	Efetuar teste unitário para a classe	Os testes foram realizados no módulo de Back-End através de aplicativo console.
27/04/07	Validar alternativas tecnológicas	Durante esta fase do processo foi sugerido estudar ANT, para automatizar alguns procedimentos para instalação do SiGES, não obtendo sucesso.

Fonte: Elaborada pelo autor.

4.2.4 FASE DE VALIDAÇÃO

Durante o desenvolvimento do estudo de caso, não houve tempo hábil para entregar e instalar formalmente o produto no cliente. Consequentemente, não houve condições de verificar defeitos em ambiente de produção, bem como do planejamento e execução das devidas correções.

4.3 AVALIAÇÃO DA APLICAÇÃO DO PAS

A aplicação do PAS no desenvolvimento do SIGES foi a primeira oportunidade de aplicação do processo com uma equipe, cliente e produto reais. Essa primeira aplicação teve por obetivo verificar se o processo estava atendendo aos seus princípios básicos: simples, didático e compatível com o meio acadêmico.

Ao final do semestre letivo, foi realizada uma reunião entre equipe de desenvolvimento, cliente e professores orientadores, com o objetivo de colher impressões e sugestões relativas à aplicação do processo. Apesar da fase de construção do PAS não ter sido realizada em sua plenitude e consequentemente o produto não ter sido finalizado, foi possível observar várias oportunidades de melhorias no processo. Foi consenso por parte de todos que o processo estava atendendo aos seus princípios fundamentais, já citados, e as melhorias deveriam acontecer.

Tabela 4.9: Atividades da Disciplina de Requisitos - Construção

Data	Atividade	Execução
25/06/2007	Início do detalhamento do caso de uso montar escala	Início da documentação do caso de uso montar escala, junto com o cliente. Nesse ponto, a equipe encontrou dificuldades no entendimento da lógica de negócio do caso de uso, devido a complexidade do mesmo.
06/07/2007	Consulta e Entrevista com o cliente.	Com o intuito de esclarecer as dúvidas surgidas na atividade anterior, foi realizada uma reunião com o cliente, visando o entendimento do caso de uso.
06/07/2007	Identificar mudanças nos requisitos	Não houve mudança nos requisitos. Iniciou-se então a disciplina de análise e projeto.

Fonte: Elaborada pelo autor.

Tabela 4.10: Atividades da Disciplina de Análise e Projeto - Construção

Data	Atividade	Execução
16/07/2007	Leitura do caso de uso	Leitura do documento de detalhamento de caso de uso.
16/07/2007	Identificar conceitos	A partir de reuniões entre cliente e equipe, foram identificados conceitos a respeito do caso de uso montar escala.
16/07/2007	Identificar operações	A partir de reuniões entre cliente e equipe, foram identificados operações para o caso de uso montar escala. Devido a complexidade do caso de uso foram necessarias varias reuniões para identificar essas operações, causando assim atrazo na implementação.
16/07/2007	Identificar Componentes	A partir de reuniões entre cliente e equipe, foram identificados componentes para o caso de uso montar escala.
16/07/2007	Elaborar Diagramas de interação de objetos	Criação do diagrama de interação de objetos do caso de uso montar escala. Devido a grande quantidade de mudanças no caso de uso, houve uma frequente mudança nesses diagramas.
18/07/2007	Alterar diagrama de classe do projeto	Ocorreram várias mudanças no diagrama de classes.
20/07/2007	Executar implementação do caso de uso.	Após a finalização das atividades de análise e projeto, deu-se início a implementação e construção das interfaces do sistema, bem como da lógica de negócio.

Fonte: Elaborada pelo autor.

Tabela 4.11: Atividades da Disciplina de Implementação e Teste - Construção

Data	**Atividade**	**Execução**
20/07/07	Planejar a Integração	O planejamento da integração dos novos casos de usos desenvolvidos foi realizado pelo engenheiro da equipe.
20/07/07	Dividir componentes entre os membros da equipe	O caso de uso foi divido em dois componentes: back-end, responsável pela lógica de negócio do sistema, e o front-end, interface entre sistema e o usuário do sistema.
20/07/07	Implementar componente	A implementação foi realizada por toda a equipe do SIGES, devido a grande complexidade lógica do caso de uso, sendo realizada em 2,5 (duas e meia) semanas.
10/08/07	Integrar o sistema	A integração do sistema foi realizada pelo gerente da equipe, com sucesso.

Fonte: Elaborada pelo autor.

Ao final da reunião, após o relato de todos a respeito da aplicação do PAS, chegou-se as seguintes oportunidades de melhoria:

1. Necessidade da disciplina de gerência de projeto;
2. Necessidade de maior ênfase na realização dos testes;
3. Revisão do processo no intuito de diminuí-lo onde for possível;
4. Estudo da real necessidade da fase de validação para projetos acadêmicos;
5. Realização de avaliação, no intuito de obter *feedback* do cliente e da equipe após entrega da iteração
6. Atividades de entrega de uma iteração (como fazer);
7. Possibilidade de sugestão por parte do PAS dos tipos de ferramentas a serem utilizadas durante o processo;
8. Definição de papéis de colaboração no desenvolvimento para o PAS;
9. Necessidade de uma ferramenta de registro das entregas e do registro do *feedback* do cliente e da equipe;

5 CONCLUSÃO E TRABALHOS FUTUROS

O PAS foi aplicado no desenvovimento do SIGES utilizando uma equipe de desenvolvimento e um cliente real. Nem todas as fases propostas pelo processo foram executadas, no entanto, um produto funcional foi gerado. A fase que deixou de ser executada, corresponde a implantação do sistema no ambiente do cliente, o que não inviabilizou a construção do produto. Mas do que um software funcionando, a aplicação do PAS proporcionou uma facilidade na condução do desenvolvimento, verificada por todos os doscentes e discentes participantes da disciplina de PDS-I, bem como o alcance do objetivo mais importante: o conhecimento da engenharia de software, adquirido por toda a equipe de desenvolvimento.

Apesar da facilidade proporcionada pela aplicação do PAS, fez ainda necessário a utilização de ferramentas auxiliares. Apesar de não poder fundamentar cientificamente que ferramentas auxiliam no trabalho em equipe, elas foram muito importantes, a medida que proporcionaram um ganho de tempo na execução de determinadas tarefas, bem como o acompanhamento do desenvolvimento, mesmo sem estar presente em sala de aula.

5.1 TRABALHOS FUTUROS

O estudo de caso realizado neste trabalho abrangeu apenas a aplicação do PAS, com o intuíto de verificar se o mesmo atende aos objetivos propostos quando de sua construção. No entanto, existem outros aspectos que podem ser desenvolvidos, visando a evolução e melhoria do processo em si.

O primeiro fator, considerado o mais importante, é a forma com que o aluno tem acesso a documentação explicativa do PAS. Após a primeira aplicação do processo, tem-se informações suficientes para a elaboração de um livro, que descreva todas as fases de execução do processo, orientado a um estudo de caso, que pode inclusive ser o mesmo deste trabalho. Além do livro, o sítio do processo, que já existe, deve ser atualizado, contendo inclusive uma referência a todos os projetos desenvolvidos utilizando o PAS.

Um outro ponto importante, é a execução de um estudo para verificação da obrigatoriedade da fase de validação. Como dito na seção 3 do capítulo 3, é nessa fase que o sistema é implantado no ambiente do cliente e validado pelo mesmo. No entanto, deixar de realizar tal fase, não implica em perda de aproveitamento por parte do aluno. Uma proposta de solução seria, realizar uma validação para cada caso de uso implementado e, só se for o caso, executar a fase de

validação, que teria exclusivamente a função de implantação do sistema no ambiente do cliente.

É também considerado um aspecto importante a definição dos papéis a serem executados pelos integrantes da equipe. Quando da construção do PAS, não foi definido quais papéis deveriam ser executados durante o desenvolvimento do sistema. Neste estudo de caso, os papéis foram definidos de acordo com os pontos fortes e fracos dos integrantes da equipe. Devido esse fato, as atividades relacionadas a teste ficaram aquém do desejado, já que são poucos os alunos que possuem uma mentalidade de "implementar e testar".

O quarto aspecto importante é a necessidade de uma disciplina relacionada com a gerência do projeto. Por muitas vezes, o professor ou gerente da equipe, não sabem que atividade deve ser executada e nem como, gerando um certo descontrole e um afastamento entre equipe, gerente e professor. A criação de uma disciplina para cobrir essa deficiência, ajudaria inclusive o gerente da equipe, seja ele professor ou aluno, na condução do processo.

Por fim, é importante a realização de uma avaliação qualitativa do processo, com o intuito de proporcionar o crescimento do processo, bem como sua adequação as situações atípicas que possam vir a surgir.

REFERÊNCIAS

AMBLER, S. W. **Modelagem ágil**: práticas eficazes para a programação extrema e o processo unificado. [S.l.]: Bookman, 2004.

BAETJER, J. H. **Software as capital**: an economic perspective on software engineering. [S.l.]: Wiley-IEEE Computer Society Press, 1997.

BECK, K. **Programação extrema explicada**: acolha as mudanças. [S.l.]: Bookman, 2004.

CANTOR, M. R. **Object-oriented project management with UML**. [S.l.]: John Wiley & Sons, 1998.

COCKBURN, A. **Agile software development**. [S.l.]: Addison-Wesley, 2002.

COCKBURN, A. **Crystal clear**: a human-powered methodology for small teams. [S.l.]: Addison-Wesley, 2005.

FALBO, R. de A. **Intregração de conhecimento em um ambiente de desenvolvimento de software**. Doutorado em Engenharia de Sistemas e Computação — Universidade Federal do Rio de Janeiro (UFRJ), 1998.

FALBO, R. de A. A experiência na definição de um processo padrão baseado no processo unificado. In: SIMPÓSIO INTERNACIONAL DE MELHORIA DE PROCESSO DE SOFTWARE, 2., 2000, São Paulo. **Anais...**. São Paulo, 2000. p. 63–75.

FILHO, W. de P. P. **Engenharia de software**: fundamentos, métodos e padrões. [S.l.]: Livros Técnicos e Científicos Editora S.A. (LTC), 2003.

GARCIA, F. P. et al. Easyprocess: um processo de desenvolvimento para uso no ambiente acadêmico. In: WEI - WORKSHOP DE EDUCAÇÃO EM COMPUTAÇÃO, 12., 2004, Salvador. **Anais...**. Salvador, 2004.

HUMPHREY, W. S. **Managing the software process**. [S.l.]: Addison Wesley Professional, 1989.

INSTITUTE, P. M. **Project managment body of knowledge guide (PMBOK)**. [S.l.]: Project Managment Institute, 2004.

ITABORAHY, A. et al. Aplicação do método scampi para avaliação do processo de gerenciamento de projetos de software numa instituição financeira. In: SIMPÓSIO INTERNACIONAL DE MELHORIA DE PROCESSO DE SOFTWARE, 7., 2005, São Paulo. **Anais...**. São Paulo, 2005.

JACOBSON, I.; BOOCH, G.; RUMBAUGH, J. **The unified software development process**. [S.l.]: Addison-Wesley, 1999.

MARTINS, V. **O processo unificado de desenvolvimento de software**. 1999. Disponível em: <http://www.pr.gov.br/batebyte/edicoes/1999/bb89/software.htm>. Acesso em: 15 nov. 2006.

MICROSOFT. **Microsoft solutions framework**. 2007. Disponível em: <http://www.microsoft.com/technet/solutionaccelerators/msf/default.mspx>. Acesso em: 30 maio 2007.

PONTES, B. P.; ALEIXO, F.; MINORA, L. A. Processo acadêmico simplificado: uma proposta de processo para o CEFET-RN/DATINF. **Holos**, v. 3, p. 74, 2006. Disponível em: <http://www.cefetrn.br/ojs/index.php/HOLOS/article/view/20/21>. Acesso em: 24 mar. 2007.

PRESSMAN, R. S. **Engenharia de software**. 6. ed. [S.l.]: McGraw-Hill, 2006. 1090 p.

RATIONAL SOFTWARE. **Rational Unified Process (RUP)**. IBM, 2007. Disponível em: <http://www.ibm.com/rational/rup>. Acesso em: 30 jan. 2007.

SAVIANI, N. **Saber escolar, currículo e didática**. [S.l.]: Autores Associados, 2003.

SCHWABER, K. **Agile project management with scrum**. [S.l.]: Microsoft Press, 2004.

SCOTT, K. **UML explained**. [S.l.]: Addison-Wesley, 2001.

SCOTT, K. **O processo unificado explicado**. [S.l.]: Bookman, 2003.

SOMMERVILLE, I. **Engenharia de software**. 6. ed. [S.l.]: Pearson Addison Wesley, 2003. 604 p.

APÊNDICE A – Modelo documento de visão

Nome do Sistema
Documento de visão
Versão x.x.x

Histórico da Revisão

Data	Versão	Descrição	Autor

A.1 INTRODUÇÃO

A.2 REQUISITOS DO CLIENTE/USUÁRIO

A.2.1 REQUISITOS FUNCIONAIS

Tabela A.1: Lista de requisitos funcionais

Cod.	Nome	Descrição

A.2.2 REQUISITOS NÃO FUNCIONAIS

Tabela A.2: Lista de requisitos não funcionais

Cod.	Descrição	Categoria	Ref. cruzada

A.2.3 DIAGRAMA DE CASOS DE USO

A.2.4 ATORES

Tabela A.3: Lista de atores

Ator	Descrição

A.2.5 CASOS DE USO

Tabela A.4: Lista de casos de uso do sistema

Caso de uso	**Descrição**

A.3 CLIENTES

A.4 DEPENDÊNCIAS

APÊNDICE B – Modelo documento de arquitetura

Nome do Sistema
Documento de Arquitetura de Software
Versão x.x.x

Histórico da Revisão

Data	Versão	Descrição	Autor

B.1 ARQUITETURA

B.1.1 VISÃO DO AMBIENTE COMPUTACIONAL E FÍSICO

B.1.2 VISÃO MACRO DOS PACOTES OU SUB-SISTEMAS

B.2 ESTILO ARQUITETURAL

B.2.1 DESCRIÇÃO DO ESTILO ARQUITETURAL

B.3 PACOTES OU SUB-SISTEMAS

B.4 ANEXOS

APÊNDICE C – Modelo documento de detalhamento de casos de uso

Nome do Sistema
Descrição do caso de uso: **Nome do Caso de Uso**
Versão x.x.x

Histórico da Revisão

Data	Versão	Descrição	Autor

C.1 RESUMO

C.2 ATORES

C.3 PRE CONDIÇÕES

C.4 POS CONDIÇÕES

C.5 FLUXOS DE ATIVIDADES

C.5.1 FLUXO PRINCIPAL

C.5.2 FLUXO ALTERNATIVO - SE FOR O CASO

C.5.3 FLUXO DE EXCEÇÃO - SE FOR O CASO

C.6 PROTÓTIPO DE INTERFACE COM O USUARIO

C.7 DIAGRAMA DE CLASSE DE ANÁLISE

C.8 DIAGRAMA DE FUNCIONALIDADES DO SISTEMA

C.9 DIAGRAMAS DE SEQÜÊNCIA

APÊNDICE D – Modelo documento do plano de testes

Nome do Sistema
Plano de Testes: **Nome do Caso de Uso**
Versão x.x.x

Histórico da Revisão

Data	Versão	Descrição	Autor

D.1 INTRODUÇÃO

D.2 PLANO DE TESTES

D.2.1 ESTRATÉGIA DE TESTE

D.2.2 PLANO

D.2.3 CASOS DE USO X CASOS DE TESTE

D.2.4 CASOS DE TESTE

D.3 AVALIAÇÃO

D.3.1 NÚMERO DE ERROS

D.3.2 COMENTÁRIOS DO TESTADOR

D.4 ANEXOS

APÊNDICE E – Modelo documento de configuração do ambiente

Nome do Sistema
Tutorial de configuração do ambiente
Versão x.x.x

Histórico da Revisão

Data	Versão	Descrição	Autor

E.1 RESUMO

E.2 SOFTWARES NECESSÁRIO

E.3 IMPORTANDO O PROJETO

E.4 CONFIGURANDO AS BIBLIOTECAS DO PROJETO

E.5 POPULANDO O BANCO DE DADOS

E.6 EXECUTANDO A APLICAÇÃO

E.7 ANEXOS

APÊNDICE F – Modelo plano de integração

Nome do Sistema
Plano de Integração: **Identificação da Iteração**
Versão x.x.x

Histórico da Revisão

Data	**Versão**	**Descrição**	**Autor**

F.1 INTRODUÇÃO

F.2 SUBSISTEMAS

F.3 BUILDS

F.4 ANEXOS