GISLAINE CAMILA LAPASINI LEAL

UMA ABORDAGEM INTEGRADA DE DESENVOLVIMENTO E TESTE DE SOFTWARE PARA EQUIPES DISTRIBUÍDAS

MARINGÁ

2010

GISLAINE CAMILA LAPASINI LEAL

UMA ABORDAGEM INTEGRADA DE DESENVOLVIMENTO E TESTE DE SOFTWARE PARA EQUIPES DISTRIBUÍDAS

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Estadual de Maringá, como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Orientadora: Profa. Dra. Elisa Hatsue Moriya Huzita

MARINGÁ

2010

Dados Internacionais de Catalogação-na-Publicação (CIP)
(Biblioteca Central - UEM, Maringá – PR., Brasil)

L435a

Leal, Gislaine Camila Lapasini
Uma abordagem integrada de desenvolvimento e teste de software para equipes distribuídas. / Gislaine Camila Lapasini Leal. -- Maringá, 2010.
xvi, 169 f. : il., figs., tabs.

Orientadora : Prof.ª Dr.ª Elisa Hatsue Moriya Huzita.
Dissertação (mestrado) - Universidade Estadual de Maringá, Programa de Pós-Graduação em Ciência da Computação, 2010.

1. Software - Desenvolvimento distribuído - Processo. 2. Software - Desenvolvimento distribuído - Teste. 3. Desenvolvimento distribuído de software. I. Huzita, Elisa Hatsue Moriya, orient. II. Universidade Estadual de Maringá. Programa de Pós-Graduação em Ciência da Computação. III. Título.

CDD 21.ed. 004.21

GISLAINE CAMILA LAPASINI LEAL

UMA ABORDAGEM INTEGRADA DE DESENVOLVIMENTO E TESTE DE SOFTWARE PARA EQUIPES DISTRIBUÍDAS

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Estadual de Maringá, como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Aprovada em 05/07/2010.

BANCA EXAMINADORA

Profa. Dra. Elisa Hatsue Moriya Huzita
Universidade Estadual de Maringá – DIN/UEM

Profa. Dra. Tania Fatima Calvi Tait
Universidade Estadual de Maringá – DIN/UEM

Prof. Dr. Paulo Cézar Stadzisz
Universidade Tecnológica Federal do Paraná – DAINF/UTFPR

Aos meus pais ...
À pessoa especial da minha vida...
À Fred Dutra (*in memorian*)...

Agradecimentos

À pessoa especial que apareceu na minha vida e me fez entender que *"Me deberías esperar y caminar a paso lento, muy lento..."*

Aos meus pais Gervásio Leal e Noeli Lapasini, pelo amor incondicional, o carinho e a força...

À professora Dra. Elisa Hatsue Moriya Huzita, minha orientadora, pela paciência e por ter depositado confiança em meu trabalho mais uma vez...

Ao professor Dr. Márcio Eduardo Delamaro (ICMC/USP) pela paciência, confiança e apoio...

Aos "irmãos"de mestrado, César Alberto da Silva e Ana Paula Chaves, por sempre estarem ao meu lado...

Aos amigos de turma, Shido, Beto, Juliana, Clara e Huff por tudo que dividimos durante essa jornada...

Ao pessoal do LabES pela recepção e pelos ensinamentos durante minha estadia em São Carlos: André Abe (Viça), Gabriel (Ceará), Fabiano, Rodrigo, Vinícius (Fofo), Kátia, Rafael (Frotinha), Adriano (Delinha), Sandro (KLB) e Lucas...

À Márcia Samed pelas infindáveis conversas, conselhos, pelos cafés no fim da tarde, pelas tristezas e alegrias que dividimos e, principalmente, pela amizade.

À Thelma Elita Colanzi pelo ombro amigo nos momentos díficeis que passei, pelas contribuições no meu trabalho, pelos conselhos, pelas atividades desportivas e, principalmente, pela amizade...

À Maria Vandete (Negavan) pela amizade, carinho e ensinamentos sobre aquilo que realmente tem valor na vida ...

Ao amigos da Engenharia de Produção, em especial: Heliana Márcia, Olívia Oiko e Elda Salomé...

Aos professores do Departamento de Informática que de diversas formas contribuíram com esse trabalho, com a minha formação acadêmica e também pessoal...

À Maria Inês Davanço por sua dedicação, carinho e simpatia...

À Capes (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior), pelo apoio financeiro...

À Deus, por ter me dado forças nos momentos de fraqueza, por colocar pessoas tão especiais na minha vida. Enfim, por ter me proporcionado todos esses motivos para agradecer...

Resumo

O Desenvolvimento Distribuído de Software (DDS) é uma estratégia de desenvolvimento que atende às necessidades da globalização no que se refere a produtividade e redução de custos. No entanto, ele acrescentou novos elementos ao processo de desenvolvimento, tais como a distância temporal, dispersão geográficae as diferenças sócio-culturais, que amplificaram alguns dos desafios e, sobretudo, acrescentaram novas exigências no que diz respeito aos processos de comunicação, coordenação e controle dos projetos. Entre essas novas demandas há a necessidade de um processo de software que ofereça apoio adequado ao desenvolvimento distribuído de software. Este trabalho apresenta uma abordagem integrada de desenvolvimento e teste de software que aborda as peculiaridades de equipes distribuídas. O objetivo da abordagem é oferecer suporte à estratégia de desenvolvimento, proporcionando uma melhor visibilidade de um projeto, melhorando a comunicação entre as equipes de desenvolvimento e teste, minimizando a ambiguidade e a dificuldade em compreender os artefatos e atividades. Esta abordagem integrada foi concebida fundamentada em quatro pilares: (i) identificar as peculiaridades do Desenvolvimento Distribuído de Software relacionadas a processos de desenvolvimento e de teste; (ii) definir os elementos necessários para compor a abordagem integrada de desenvolvimento e teste para apoiar equipes distribuídas; (iii) descrever e especificar os *workflows*, artefatos e papéis da abordagem; e, (iv) representar a abordagem de forma adequada para possibilitar a comunicação e entendimento efetivo da mesma. A abordagem foi avaliada, seguindo os princípios da Engenharia Experimental, por meio da condução de um estudo de viabilidade.

Abstract

The Distributed Software Development (DSD) is a development strategy that meets the globalizations needs in relation to productivity and cost reduction. However, it added new elements to the development process, such as the temporal distance, geographical dispersion and the socio-cultural differences, which amplified some challenges and, especially, added new requirements in relation to the communication process, coordination and control of projects. Among these new demands there is the necessity of a software process that provides suited support to the distributed software development. This work presents an integrated approach of software development and test that considers the peculiarities of distributed teams. The goal of the approach is to offer support to the strategy of development, providing a better visibility of a project, improving the communication between the development and test teams, minimizing the ambiguity and difficulty in understanding the artifacts and activities. This integrated approach was conceived based on four pillars: (i) to identify the peculiarities of Distributed Software Development related to development and test processes, (ii) to define the necessary elements to compose the integrated approach of development and test to support the distributed teams, (iii) to describe and specify the workflows, artifacts, and roles of the approach, and (iv) to represent appropriately the approach to enable the effective communication and understanding of it. The approach was evaluated, using the Experimental Engineering principles, by conducting a feasibility study.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Siglas

ADDS: Ambiente de Desenvolvimento Distribuído de Software
AUP: *Agile Unified Process*
CIM: *Computational Independent Model*
CITM: *Computational Independent Testing Models*
CMMI: *Capability Maturity Model Integration*
DDS: Desenvolvimento Distribuído de Software
DiSEN: *Distributed Software Engineering Environment*
GSD: *Global Software Development*
LAGPRO: *Local Agile Game-based Process*
MDA: *Model Driven Architecture*
MDD: *Model Driven Development*
MDT: *Model Driven Test*
OCL: *Object Constraint Language*
OMG: *Object Management Group*
PIM: *Platform Independent Model*
PITM: *Plataform Independent Testing Models*
PML: *Process Modelling Language*
PSM: *Platform Specific Model - PSM*
RUP: *Rational Unified Process*
SPEM: *Software Process Engineering Metamodel Specification*
SUT: *System Under Test*
TBM: Teste Baseado em Modelos
TDD: *Test-Driven Development*
TDF: Técnica de Descrição Formal
TOTEM: *Testing of Object-Oriented Software SysTEms with the UML*
UML: *Unified Modelling Language*
UPM: *Unified Process Model*
U2TP: *UML 2.0 Testing Profile*
XP: *eXtreme Programming*

Capítulo

1

Introdução

1.1 Considerações Iniciais

O processo de software é definido como um conjunto ordenado de atividades para o gerenciamento, desenvolvimento e manutenção de software, e deve estar alinhado com as condições organizacionais (Fuggetta, 2000).

O Desenvolvimento Distribuído de Software (DDS) é uma abordagem para o desenvolvimento dos projetos de software que vem ao encontro das necessidades da globalização, que são: aumento de produtividade, melhoria de qualidade e redução de custos (Herbsleb et al., 2000). Essa nova configuração adicionou ao desenvolvimento de software desafios relacionados às diferenças culturais, dispersões geográficas, coordenação e controle, comunicação e espírito de equipe, os quais intensificam alguns dos problemas enfrentados durante o ciclo de vida do projeto.

No cenário atual, vivenciamos uma crescente necessidade de software com qualidade. Diante disso, muitas formas de melhoria dos processos de desenvolvimento de software são utilizadas e diversos artefatos e ferramentas buscam melhorar o desenvolvimento, como a utilização de métodos formais para descrever especificações, *design* e testes (Abdurazik, 2000). Analisando os trabalhos apresentados no Capítulo 3 foi possível observar que os processos de engenharia de software são, geralmente, voltados para o desenvolvimento de projetos coalocados, não atendendo totalmente às necessidades do desenvolvimento distribuído. E os processos que foram desenvolvidos especificamente para o contexto de desenvolvimento distribuídos não contemplam os elementos básicos de um processo

e as peculiaridades dessa abordagem. Assim, os desafios de comunicação, coordenação e controle gerados pelo DDS, decorrentes das distâncias física e temporal, demandam processos de desenvolvimento de software que sejam adequados para serem utilizados por equipes distribuídas.

O objetivo deste trabalho é apresentar uma abordagem integrada de desenvolvimento e teste de software que apoie o desenvolvimento distribuído de projetos, contemplando as suas peculiaridades e, com isso, melhorar a comunicação entre as equipes, minimizar a ambiguidade nos artefatos e fornecer, aos envolvidos, visibilidade adequada em relação aos artefatos e atividades.

1.2 Contexto e Motivação

Nos últimos anos pode-se observar que algumas organizações descentralizaram suas atividades de desenvolvimento de software em busca de vantagens competitivas, tais como: acesso a mão-de-obra qualificada, redução de custos, avanços na infraestrutura e outros. Essa distribuição de atividades em um mesmo país ou até mesmo em países diferentes, caracterizam o DDS.

O principal objetivo dessa descentralização consiste em otimizar os recursos para desenvolver produtos de maior qualidade e a um custo menor. No entanto, essa dispersão adicionou novos desafios ao desenvolvimento relacionados à comunicação, coordenação e controle, os quais podem afetar negativamente a produtividade e, por conseguinte, a qualidade do software. Esses fatores influenciam a maneira pela qual o software é projetado, desenvolvido, testado e entregue aos clientes, afetando assim os estágios correspondentes do ciclo de vida do software. Segundo Damian e Lanubile (2004), para minimizar esses efeitos e alcançar níveis mais elevados de produtividade, são necessárias novas tecnologias, processos e métodos compatíveis com a abordagem de desenvolvimento distribuído.

Em Jiménez et al. (2009) é possível observar a evolução dessa abordagem de desenvolvimento em função do aumento de trabalhos disponíveis na literatura. Além disso, nota-se que os maiores esforços estão centrados em processos organizacionais, os quais tratam dos recursos humanos, gestão organizacional, alinhamento de infraestrutura e gestão de projetos. Isto evidencia uma lacuna em relação ao nível de projetos e aspectos técnicos o que requer novos processos e ferramentas.

Diante deste cenário, este trabalho apresenta a especificação de uma abordagem integrada de desenvolvimento e teste de software compatível com as necessidades das equipes distribuídas.

1.2.1 Descrição do Problema

Os processos de engenharia de software possuem artefatos, responsabilidades e atividades bem definidas, possibilitando adequação às necessidades específicas. No entanto, estes processos são, geralmente, voltados para o desenvolvimento de projetos coalocados, não considerando as peculiaridades de coordenação, controle e comunicação do desenvolvimento com equipes geograficamente dispersas. Rocha et al. (2008) enfatizam a necessidade de processos adequados a essa estratégia de desenvolvimento. De acordo com (Audy e Prikladnicki, 2008), em um ambiente de desenvolvimento distribuído, é fundamental um processo de desenvolvimento comum à equipe, visto que auxilia diretamente na sincronização, fornecendo a todos os membros da equipe uma nomenclatura comum de tarefas e atividades, e um conjunto comum de expectativas aos elementos envolvidos no processo. Diante de tal cenário observou-se a necessidade de desenvolver uma abordagem que forneça suporte adequado ao desenvolvimento distribuído.

1.2.2 Solução Proposta

A abordagem integrada de desenvolvimento e teste de software, apresentada nesse trabalho, vem ao encontro das necessidades do desenvolvimento distribuído. Essa abordagem objetiva oferecer suporte aos processos de comunicação, coordenação e controle fornecendo uma nomenclatura comum aos envolvidos no que se refere às disciplinas, papéis, atividades e artefatos. Além disso, objetiva-se reduzir a ambiguidade e imprecisão dos artefatos com o uso de uma especificação mais rigorosa e oferecer visibilidade sobre as disciplinas em execução, os envolvidos e suas responsabilidades.

1.3 Objetivos

Para oferecer apoio ao desenvolvimento distribuído de software, melhorando a comunicação entre as equipes de desenvolvimento e teste e minimizando a ambiguidade nos artefatos, o objetivo geral desta dissertação é apresentar uma abordagem integrada de desenvolvimento e teste de software para equipes distribuídas. Como objetivos específicos, têm-se:

- identificar as peculiaridades do Desenvolvimento Distribuído de Software relacionadas a processos de desenvolvimento e de teste;
- definir os elementos necessários para compor a abordagem integrada de desenvolvimento e teste para apoiar equipes distribuídas;

- descrever e especificar os *workflows*, artefatos e papéis da abordagem;
- representar a abordagem proposta de forma adequada para possibilitar a comunicação e entendimento efetivo da mesma.

1.4 Metodologia

A pesquisa conduzida neste trabalho se caracteriza como sendo de base qualitativa e do tipo exploratória, sendo uma adaptação da metodologia proposta por Mafra (2006), que extende o trabalho de Shull et al. (2001), como pode ser visto na Figura 1.1. Essa metodologia objetiva possibilitar o desenvolvimento e a maturação de tecnologias desde sua concepção acadêmica até sua inserção na indústria, sendo composta por duas fases, Definição e Refinamento.

Durante a fase de Definição são conduzidas revisões sistemáticas para identificar evidências disponíveis na literatura e deste modo minimizar dificuldades e incertezas no processo de definição. Na fase de Refinamento, o foco é coletar evidências, pontos fortes e fracos sobre a aplicação da tecnologia e, baseada nessas evidências, propor alterações e melhorias. Para isso, é proposta a condução de estudos de viabilidade, estudos observacionais e estudos de caso, tanto no contexto de um ciclo de vida completo da tecnologia, como no contexto industrial (Shull et al., 2001).

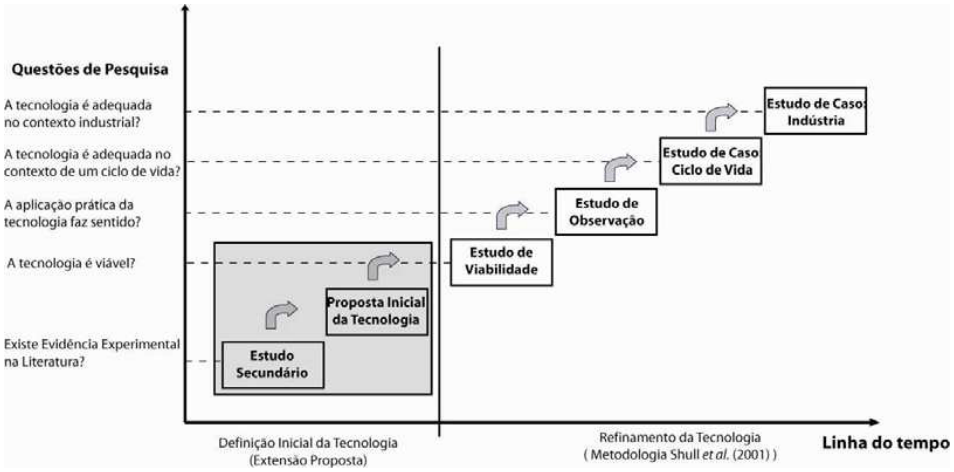

Figura 1.1: Metodologia proposta por Mafra (2006).

A metodologia utilizada no desenvolvimento deste trabalho é composta por cinco fases, conforme ilustrado na Figura 1.2. As três primeiras fases correspondem à fase de Definição

e a quarta fase a de Refinamento, apresentada por Mafra (2006). A seguir a metodologia é descrita.

Na primeira fase foi realizada uma revisão inicial da literatura, que teve como objetivo formar um referencial teórico consistente para a continuidade do estudo e visualizar o estado da arte. Esta etapa envolveu estudos sobre desenvolvimento distribuído de software, processos de desenvolvimento e teste de software e notações para modelagem de processos.

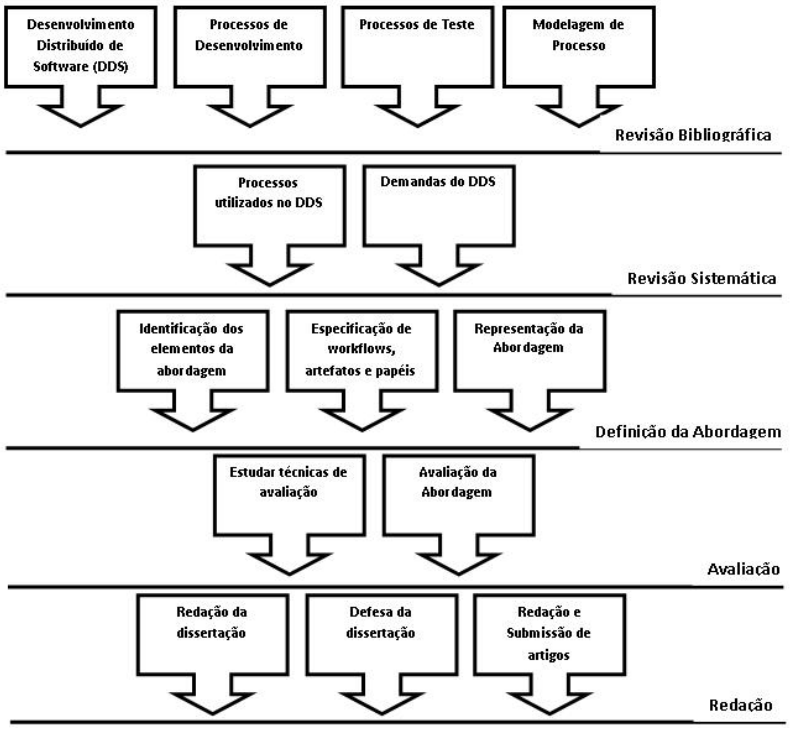

Figura 1.2: Metodologia de trabalho.

A segunda fase consistiu na condução de uma revisão sistemática, seguindo o modelo de protocolo apresentado em (Biolchini et al., 2005). A revisão sistemática foi conduzida para ampliar a cobertura da revisão de literatura inicial e o objetivo principal foi a identicação de trabalhos que abordam processos de software utilizados no desenvolvimento com equipes distribuídas. Além disso, a revisão também visou a identificação das necessidades dessa abordagem de desenvolvimento no que se refere ao processo de desenvolvimento. A revisão sistemática possibilitou identificar os processos de desenvolvimento de software que vem sendo utilizados com equipes distribuídas. Os estudos selecionados foram, na maior parte, referentes aos problemas (diversidade cultural, dispersões geográficas, coordenação e comunicação) encontrados quando da adoção de desenvolvimento distribuído, realçando

a necessidade da definição de um processo que contemple as características dessa nova abordagem de desenvolvimento.

Na terceira fase, Definição da Abordagem, foi possível analisar e definir, com base nas etapas anteriores, os elementos necessários para uma abordagem integrada de desenvolvimento e teste de software que contemple os aspectos relevantes em desenvolvimento distribuído. Após a identificação, os elementos da abordagem foram descritos e especificados de acordo com a técnica de modelagem definida.

A quarta fase, Avaliação, consistiu na condução de um estudo de viabilidade o qual teve por objetivo criar um corpo de conhecimento sobre a aplicação da tecnologia e não encontrar uma resposta definitiva. Nesse sentido, foi possível ao pesquisador avaliar se a aplicação da tecnologia é viável, ou seja, se atende de forma adequada aos objetivos inicialmente definidos, de forma a justificar (ou não) a continuação da pesquisa. Além disso, o corpo de conhecimento construído fornece subsídios para o refinamento da tecnologia e a geração de novas hipóteses sobre sua aplicação a serem investigadas em estudos posteriores (Shull et al., 2001). As demais etapas do Refinamento não foram contempladas no escopo deste trabalho devido ao tempo disponível.

Por fim, a quinta fase, Redação, consistiu em produzir este documento de dissertação, na escrita e submissão de artigos relacionados a este trabalho e na defesa da dissertação.

1.5 Organização do Trabalho

Neste capítulo foram descritos os objetivos deste trabalho, embasados nos problemas encontrados durante o desenvolvimento distribuído de software no que se refere a processos de desenvolvimento e teste de software. Além disso, foi apresentado o contexto, a motivação e a metodologia que norteou o desenvolvimento do mesmo.

O restante deste trabalho encontra-se organizado da seguinte forma:

- Capítulo 2: apresenta os conceitos relevantes que embasaram o desenvolvimento deste trabalho, sendo eles: desenvolvimento distribuído de software, processo de desenvolvimento, norma ISO/IEC 12207, técnicas de modelagem de processos, teste de software, teste baseado em modelos, processo de teste, norma IEEE 829-1998, metodologia de geração de casos de testes e perfil UML de teste.

- Capítulo 3: descreve as características dos processos de desenvolvimento e teste de software que deram subsídios para o desenvolvimento deste trabalho. Além disso, são apresentados alguns critérios de comparação e tais processos são analisados considerando as demandas do desenvolvimento com equipes distribuídas.

- Capítulo 4: a abordagem proposta é descrita em termos de disciplinas, atividades, papéis e artefatos. Além disso, é apresentada a modelagem seguindo o metadomodelo SPEM.

- Capítulo 5: ilustra um exemplo de aplicação da abordagem proposta em um projeto de desenvolvimento de uma aplicação web para gerenciar o processo de submissão e avaliação de artigos em eventos científicos.

- Capítulo 6: apresenta o estudo viabilidade conduzido para coletar os pontos fortes e fracos da abordagem proposta.

- Capítulo 7: discute as contribuições e limitações do trabalho e apresenta os trabalhos futuros.

Capítulo

2

Revisão Bibliográfica

Neste capítulo são apresentados os conceitos relevantes que deram subsídios para o desenvolvimento deste trabalho, sendo eles: Desenvolvimento Distribuído de Software, processos de desenvolvimento, técnicas de modelagem de processo de software, normas de referência, teste de software e engenharia experimental.

2.1 Desenvolvimento Distribuído de Software

A crescente globalização do ambiente de negócios e da economia, juntamente com o avanço tecnológico, alta competitividade, aumento do tamanho das equipes, dificuldade em reunir especialistas e aumento da complexidade do software, têm afetado, diretamente, o desenvolvimento de software. Em busca de vantagem competitiva, diversas organizações optaram por distribuir o processo de desenvolvimento de software caracterizando o Desenvolvimento Distribuído de Software (DDS) (Huzita et al., 2008).

O DDS pode ser definido como o desenvolvimento de software que utiliza equipes em várias localizações geográficas, com o envolvimento de pessoas de diferentes nacionalidades e diferentes culturas organizacionais. Segundo Carmel (1999), os projetos de DDS consistem de times trabalhando juntos para atingir objetivos comuns em um mesmo projeto, porém dispersos geograficamente. Quando a distribuição desses times atinge dimensões globais, o DDS é denominado Desenvolvimento Global de Software (GSD - *Global Software Development*).

Em um ambiente de DDS a distribuição pode assumir algumas configurações, de acordo com a distância entre as equipes e as organizações envolvidas no projeto. Em relação à distribuição geográfica das unidades envolvidas em um projeto, quando elas se localizam em mais de um país denomina-se distribuição *offshore*, se todas se encontram no mesmo país tem-se a distribuição *onshore*. Considerando a relação estabelecida entre as empresas, tem-se o *outsourcing*, cenário em que a empresa delega o controle sobre uma ou mais atividades para uma empresa externa à quem contratou o serviço, e o *insourcing*, em que as empresas criam os seus próprios centros de desenvolvimento de software. Dessas configurações de distribuição surgem quatro modelos de negócios, que são:

- *onshore insourcing* ou demanda doméstica interna: neste modelo de negócio existe um departamento na própria empresa, ou uma subsidiária no mesmo país (*onshore*), que provê serviço de desenvolvimento de software, através de projetos internos (*insourcing*);

- *onshore outsourcing* ou *outsourcing*: este modelo de negócio indica a contratação de uma terceira empresa (*outsourcing*) para o desenvolvimento de determinados serviços ou produtos de software para uma empresa. A empresa terceira está localizada no mesmo país da empresa contratante (*onshore*);

- *offshore outsourcing* ou *offshoring*: este modelo de negócio indica contratação de uma empresa terceira (*outsourcing*) para o desenvolvimento de determinados serviços ou produtos de software, sendo que a empresa terceirizada está, necessariamente, localizada em um país diferente da contratante (*offshore*);

- *offshore insourcing* ou *internal offshoring*: este último modelo de negócio indica a criação de uma subsidiária da própria empresa para prover serviços de desenvolvimento de software (*insourcing*). Esta subsidiária está necessariamente localizada em um país diferente da matriz da empresa, ou empresa contratante (*offshore*).

Essa configuração de desenvolvimeto de software, o DDS, acrescentou novos fatores ao processo, tais como, distância temporal, dispersão geográfica, diferenças sócio-culturais, os quais amplificaram alguns dos desafios existentes na área de engenharia de software e adicionaram novas demandas que desafiam os processos de comunicação, coordenação e controle dos projetos (Layman et al., 2006). Diversos trabalhos ((Damian, 2002), (Herbsleb et al., 2000), (Mockus e Herbsleb, 2001), (Sangwan et al., 2006)) descrevem tais desafios, os quais podem ser relacionados a fatores técnicos (problemas de conectividade em rede e diferenças entre os ambientes de desenvolvimento e teste) e não técnicos (confiança, comunicação, conflitos e cultura).

De acordo com Mockus e Herbsleb (2001) e Damian (2002), os principais desafios encontrados no DDS estão relacionados às diferenças culturais, dispersões geográficas, coordenação e controle e comunicação. A diversidade cultural implica dificuldades de compreensão da linguagem natural utilizada nos documentos de requisitos e a convergência entre diferentes interesses (Layman et al., 2006). E as distâncias geográficas agravam os problemas de coordenação e controle, de forma direta ou indireta, por meio dos efeitos negativos que causam na comunicação (Hargreaves e Damian, 2004).

A dispersão geográfica dificulta a possibilidade de se estabelecer e manter relacionamentos interpessoais, os quais são essenciais para a construção de um espírito de equipe. Além disso, enfrentam dificuldades para chegar a um consenso sobre as práticas de comunicação e processo de desenvolvimento que devem ser adotados no projeto.

A comunicação é o fator de mediação que afeta diretamente a coordenação e o controle. Ela representa a troca de informações não ambíguas e completas, isto é, o emissor e o receptor conseguem chegar a um entendimento comum. Segundo Herbsleb et al. (2000), a comunicação informal exerce um papel chave no sucesso das equipes distribuídas, pois é considerada o componente essencial de todas as práticas de colaboração no processo de desenvolvimento de software. A comunicação ineficiente pode resultar em um baixo nível de confiança entre as equipes e na perda da visibilidade sobre o andamento dos trabalhos (Layman et al., 2006).

A coordenação consiste no ato de integrar cada atividade com cada unidade organizacional. A condução da integração, geralmente, requer comunicação intensa e constante. O controle é o processo de adesão aos objetivos, políticas, padrões e níveis de qualidade. Mockus e Herbsleb (2001) relatam que as questões de coordenação e controle se tornam um problema quando as equipes distribuídas utilizam processos diferentes e ressaltam a necessidade de um processo padrão. A Seção 2.2 descreve os conceitos de processo e destaca sua importância.

2.2 Processo de Desenvolvimento de Software

Um processo de software pode ser definido como um conjunto coerente de políticas, estruturas organizacionais, tecnologias, procedimentos e artefatos necessários para conceber, desenvolver, implantar e manter um produto de software (Fuggetta, 2000). Envolve uma série de etapas constituídas por um conjunto de atividades, métodos, práticas e tecnologias que são utilizadas desde o desenvolvimento até a manutenção do software e produtos relacionados.

Um processo pode ser imaturo e utilizar uma abordagem *ad hoc* para organizar o trabalho, ou maduro, sendo caracterizado por um método padronizado e documentado, que já tenha sido utilizada em projetos similares ou, que seja, habitualmente, adotada em uma organização.

Segundo Berger (2003), o uso de um processo padrão permite aos gerentes de projeto definir planos em conformidade com os padrões de qualidade e procedimentos da organização. Uma equipe de desenvolvimento que trabalha sem um processo bem definido acaba funcionando de maneira *ad hoc*, o que compromete o seu sucesso, criando uma situação insustentável. Por outro lado, organizações maduras empregando um processo bem definido podem desenvolver sistemas complexos de maneira consistente e previsível, independente de quem o produziu (Kruchten, 2003).

Além disso, o processo deve possibilitar a melhoria da qualidade de serviço, de engenharia e de projeto; a diminuição de custos por meio do aumento da previsibilidade e capacidade de mitigar riscos, bem como a melhoria da eficiência e produtividade da organização.

2.2.1 Norma ISO/IEC 12207

A norma ISO/IEC 12207 propõe uma arquitetura de alto nível para o ciclo de vida de software por meio da definição de processos, atividades e tarefas que podem ser aplicadas na aquisição, no fornecimento, no desenvolvimento, na operação e na manutenção de software. Cada processo é composto por atividades, e cada uma destas possui um grupo de tarefas associadas.

A ISO/IEC 12207 define 17 processos, os quais estão classificados em: processos fundamentais, processos de apoio e processos organizacionais. Os processos estabelecidos pela norma podem ser visualizados na Figura 2.1.

Os processos fundamentais compreendem os processos envolvidos na execução do desenvolvimento, operação e manutenção do software durante o ciclo de vida, além da contratação entre cliente e fornecedor. Os processos de apoio colaboram com o sucesso e a qualidade do projeto de software. Os processos organizacionais são empregados por uma organização para estabelecer e implementar uma estrutura constituída pelos processos do ciclo de vida e pelo pessoal envolvido no desenvolvimento de software.

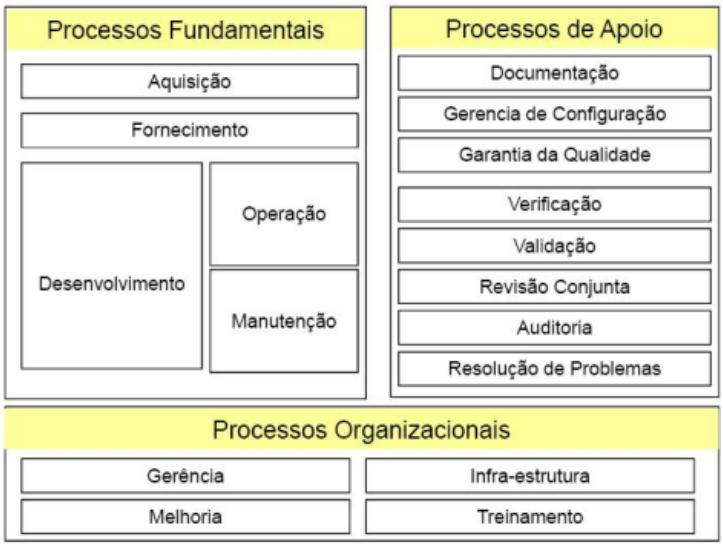

Figura 2.1: Processos da norma ISO/IEC 12207.

2.3 Técnicas de Modelagem de Processo de Software

O modelo de um processo de software é a representação formal dos elementos e características envolvidos neste processo. Segundo Ben-Shaul e Kaiser (1994), um modelo de processo deve especificar os pré-requisitos e consequências de cada tarefa, bem como a sincronização entre as mesmas.

A modelagem de um processo de software deve ser conduzida de modo a possibilitar o entendimento e a padronização do processo. Humphrey e Kellner (1989) ressaltam que a representação do processo possibilita a comunicação e o entendimento efetivo do processo, facilita o reuso, apoia a evolução, facilita o gerenciamento do processo e é importante na avaliação, evolução e melhoria contínua do processo.

A estrutura de um modelo de processo de software é, geralmente, composta por quatro elementos:

- Ator: representa as entidades que executam um processo ou assumem um papel durante a execução de uma tarefa;
- Papel: grupo de responsabilidades que executam uma atividade específica do processo de software;
- Artefato: porção representativa de informação do produto que resulta de uma atividade e pode ser utilizado posteriormente como matéria-prima para gerar novos artefatos;

- Atividade: consome artefatos (de entrada) e gera novos artefatos (de saída).

A modelagem de processos de software iniciou-se com o uso de técnicas de análise de sistemas, como a utilização de diagramas do paradigma estruturado para representar o processo. Posteriormente, surgiram as linguagens de modelagem de processos (PML - *Process Modelling Language*), que agregam os diversos elementos (ator, papel, artefato e atividade) utilizados nas formas de modelagem anteriores.

Uma PML é uma descrição computacional, que pode ser formal, semi-formal ou informal, de um processo externo. Ela expressa os processos de produção do software na forma de um modelo de processo.

Em 2000, a OMG (*Object Management Group*) apresentou o UPM (*Unified Process Model*) como uma proposta de unificação entre as diferentes metodologias para modelagem de processos. A evolução do UPM deu origem ao SPEM (*Software Process Engineering Metamodel Specification*), um metamodelo proposto pela OMG para a descrição de um processo concreto de desenvolvimento de software ou uma família relacionada de processos de desenvolvimento de software. Sua notação é baseada na UML e, por este motivo, oferece para a modelagem de processo os mesmos diagramas que são usados para modelar software.

Os principais elementos estruturais do SPEM, para a descrição de um processo, e seus relacionamentos são mostrados na Figura 2.2. Um Produto de Trabalho, também chamado de Artefato, é toda informação produzida, consumida ou modificada por um processo. Um Tipo de Produto de Trabalho descreve uma categoria para os artefatos. A Definição de Trabalho é um tipo de operação que descreve o trabalho executado em um processo. Atividade é a principal subclasse da Definição de Trabalho e descreve uma parte de trabalho executado por um Papel no Processo. Uma Atividade pode ser constituída por um conjunto de elementos atômicos, os Passos. O Executor no Processo define um executor para um conjunto de Definições de Trabalho em um processo e apresenta uma especialização chamada Papel no Processo, que define responsabilidades sobre artefatos específicos e define os papéis que executam ou auxiliam em atividades específicas.

Em um processo, as atividades com temas comuns podem ser agrupadas em Disciplinas. A OMG define alguns elementos que auxiliam na definição de como o processo será executado. Na Figura 2.3 são ilustrados os elementos que compõem o ciclo de vida de um processo.

Sob a perspectiva da execução, um processo é visto como uma colaboração entre papéis para alcançar determinada meta ou objetivo. Para guiar esta execução, pode-se considerar as restrições para ser a ordem em que as atividades devem ser executadas, chamadas de pré-condições. Fase é uma especialização da Definição de Trabalho tal que

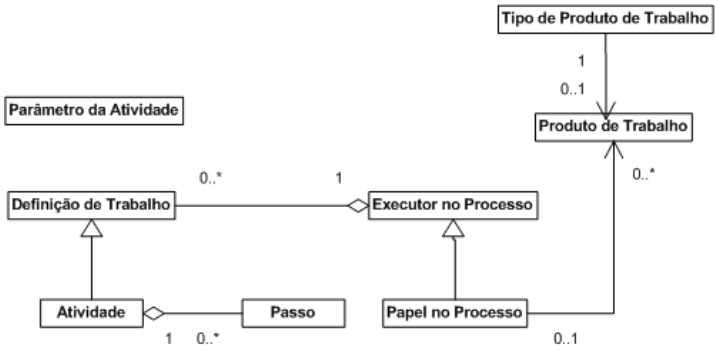

Figura 2.2: Elementos estruturais de um processo (OMG, 2005).

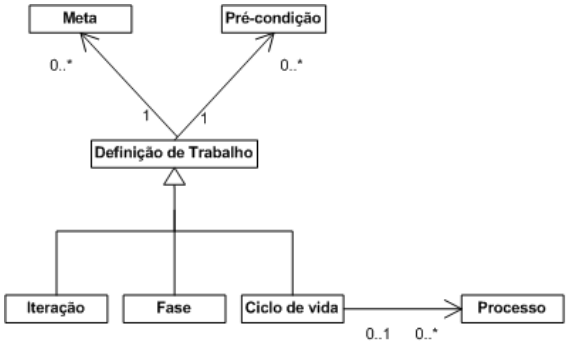

Figura 2.3: Ciclo de vida de um processo (OMG, 2005).

sua pré-condição define os critérios de entrada da fase. Os critérios de saída da fase são suas metas, os chamados *milestones*. Ciclo de Vida é definido como uma sequência de Fases que buscam uma meta específica. Iteração é uma Definição de Trabalho com *milestones* menores.

Alguns elementos do SPEM podem ser representados graficamente por meio de ícones (Tabela 2.1), outros existem apenas como metainformação, não possuindo uma representação gráfica.

Tabela 2.1: Elementos do SPEM.

Estereótipo	**Descrição**	**Notação**
Produto de Trabalho	Descrição de um pedaço de informação gerada ou consumida pelas atividades do processo, como por exemplo: modelos, planos, códigos executáveis, documentos, entre outros.	
Definição de Trabalho	Descreve a execução, as operações desempenhadas e as transformações realizadas, por papéis, em um Produto de Trabalho. Alguns exemplos de Definição de Trabalho são: atividade, iteração, fase e ciclo de vida.	
Orientação	Apresenta informações adicionais, tais como: *templates*, técnicas, procedimentos, padrões e exemplos.	
Atividade	Descreve "o que"um Papel executa no Processo.	
Papel no Processo	Descreve os papéis, responsabilidades e competências de um indivíduo que realiza Atividades dentro de um Processo.	
Executor no Processo	Descreve os proprietários das Definições de Trabalho. São usados para Definições de Trabalho que não podem associar-se com os Papéis no Processo, tais como um Ciclo de Vida ou uma Fase.	
Disciplina	Agrupamento de atividades comuns de um processo, como por exemplo: requisitos, análise, projeto e gerenciamento de configuração entre outras.	
Fase	Definição de Trabalho de alto nível, limitada por um *milestone*.	
Processo	Descrição completa do processo, em termos de Executores do Processo, Papéis no Processo, Definições de Trabalho, Produtos de Trabalho e Orientações associadas	
Documento	Tipo de Produto de Trabalho.	
Modelo UML	Tipo de Produto de Trabalho	

2.4 Teste de Software

O crescimento do uso de software e o aumento de complexidade, tamanho, heterogeneidade, autonomia, distribuição física e dinamismo de sistemas computacionais afetam diretamente a qualidade destes sistemas. Neste cenário, o teste de software adquire cada vez mais importância, pois tem como objetivo principal detectar a presença de defeitos o mais cedo possível, não somente no código-fonte, mas também nos documentos de requisitos e projeto. Visando atingir este objetivo, o teste deve ser planejado e projetado de modo a definir formas de execução do programa que tenha uma alta probabilidade de revelar defeitos no software (Myers, 2001).

Como objetivo secundário o teste fornece uma boa indicação de confiabilidade e alguns indícios da qualidade do software como um todo. A obtenção desta indicação é possível porque o teste, quando não detecta a presença de defeitos, demonstra que as funções do software estão, aparentemente, funcionando de acordo com as especificações e que os requisitos de desempenho foram, aparentemente, cumpridos.

A atividade de testes contribui para o aumento da qualidade do produto final. No entanto, assim como diversas outras atividades do processo de desenvolvimento de software, possui problemas que precisam ser evitados durante a sua realização para garantir o seu sucesso. Mats (2001) destaca os cinco principais problemas associados aos testes em software, sendo eles:

- atrasos no cronograma do projeto, impossibilitando a equipe de completar os testes planejados devido à redução de recursos e tempo;
- carência na rastreabilidade de casos e procedimentos de teste entre diferentes versões do software, o que dificulta a reutilização;
- teste manual ou não-padronizado, resultando em um grande esforço a cada início de uma nova atividade de teste;
- incerteza sobre o que está sendo testado, devido à falta de definição dos objetivos e escopo para as atividades de teste;
- ausência de critérios para a seleção do conjunto de casos e procedimentos de testes.

Segundo Juristo et al. (2004), teste de software é considerada uma das práticas mais custosas do processo de desenvolvimento e desta forma, necessita um bom gerenciamento a fim de evitar desperdícios de recursos e atrasos no cronograma, dentre outras possibilidades. Com isso, a realização de testes em software requer recursos adequados, e a

utilização efetiva desses recursos necessita de um bom planejamento e controle (McGregor e Sykes, 2001).

2.4.1 Conceitos

Essa seção descreve os principais termos e expressões utilizados na área de teste de software, sendo eles (Delamaro et al., 2007):

- Critério de teste: método ou diretriz que serve para direcionar a atividade de teste e/ou tomar decisões relativas ao teste. Um critério de teste define um conjunto de condições que devem ser utilizadas na atividade de teste;
- Sistema em teste (SUT - *System Under Test*): parte ou todo sistema, subsistema ou componente em teste, que funciona como uma caixa-preta que somente pode ser exercitada pela estrutura de teste por meio das operações disponíveis em sua interface pública;
- Cenário de teste: coleção de casos de teste reunida com a configuração de teste. Os casos de teste são executados com base no cenário de teste;
- Ambiente de teste: descrição do ambiente de hardware e software no qual os testes são executados e a descrição de qualquer software com que o sistema em teste interage quando testado usando dispositivos de teste ou *stubs*, que são esqueletos ou implementações com propósito especial de um módulo de software;
- Conjunto de teste: coleção de um ou mais casos de teste de um sistema em teste. O caso de teste é o conjunto de dados de entrada, condições de execução e resultados esperados elaborados para atingir um objetivo específico de teste, tal como exercitar um fluxo do programa ou verificar a conformidade com um requisito específico;
- Objetivo de teste: conjunto de características de um software a serem medidas em condições específicas pela comparação do comportamento atual com o comportamento requerido descrito na documentação do software;
- Dado de teste: dado enviado ao sistema em teste. Frequentemente, corresponde ao dado de entrada fornecido à interface pública do sistema em teste durante a execução do caso de teste;
- Oráculo de teste: mecanismo usado para gerar o resultado esperado de um caso de teste. Este resultado é confrontado com o resultado real obtido pela execução do

sistema em teste. O resultado observado é o dado que reflete a reação do sistema em teste a um dado de teste. Corresponde comumente ao dado de saída (ou retorno) de uma função em teste presente na interface pública de um sistema.

- Laudo de teste: registro da interação resultante da execução de um caso de teste. Está associado a um veredito que indica o grau de conformidade do sistema em teste com o objetivo definido por um caso de teste. O veredito é a sentença conferida ao sistema em teste indicando a sua corretude ou não.

Existem várias técnicas de teste que podem ser utilizadas para a seleção de subconjuntos de dados de teste. Cada uma destas técnicas possui características próprias que tornam sua aplicação mais indicada a diferentes etapas de teste.

As técnicas de teste são agrupadas em classes de técnicas similares, dependendo de suas características. As principais técnicas de teste de software são: teste estrutural, teste funcional e teste baseado em erros. As aplicações dessas técnicas detectam a presença de defeitos de categorias distintas, uma vez que elas contemplam diferentes perspectivas do software. Deste modo, impõe-se a necessidade de se estabelecer uma estratégia de teste que contemple as vantagens e os aspectos complementares de cada uma das classes, levando a uma atividade de teste de boa qualidade, eficaz e de baixo custo.

O teste estrutural ou teste de caixa branca, estabelece os elementos requeridos com base na implementação, solicitando a execução de partes e/ou componentes elementares do programa. Essa técnica verifica a estrutura interna do programa.

O teste funcional, conhecido como teste de caixa preta ou teste comportamental, estabelece os elementos requeridos com base na especificação, sem se preocupar com os detalhes da implementação. Por meio da realização do teste funcional, o software ou o sistema é tratado como uma caixa preta. Ele é executado com determinadas entradas e suas saídas são verificadas em relação ao comportamento definido por meio da especificação de software. O testador deve estar preocupado somente com a funcionalidade e com características do software.

O teste baseado em erros utiliza informações sobre os tipos de erros mais frequentes para derivar os requisitos de teste.

2.4.2 Teste Baseado em Modelos

O Teste Baseado em Modelos (TBM) é uma técnica de teste funcional que utiliza informações sobre o comportamento funcional do software para a realização do teste. Esse

comportamento é descrito por meio de um modelo, denominado modelo comportamental, o qual é construído a partir dos requisitos funcionais do software.

Os modelos comportamentais determinam, basicamente, quais são as possíveis ações durante a execução de um software e quais são as saídas esperadas. Segundo Dalal et al. (1999), o desenvolvimento de especificações na forma de modelos, mesmo se realizado em fase avançada no processo de software, é um meio efetivo de: descobrir defeitos no sistema, já que muitos se tornam visíveis simplesmente pela própria modelagem; definir rapidamente a base para os cenários de uso do sistema; e preservar esse investimento para versões futuras ou sistemas similares.

Offutt et al. (2003), destacam que a modelagem é um meio muito econômico para se capturar o conhecimento sobre um sistema e depois reutilizá-lo à medida que o sistema cresce. Um benefício em especificar os requisitos do sistema na forma de um modelo comportamental e usar esse modelo no teste de software é que os casos de teste podem ser criados no início do processo de desenvolvimento do software e estarem prontos para execução antes que o programa esteja completamente implementado. Adicionalmente, o testador geralmente encontrará inconsistências e ambiguidades nos modelos quando os testes forem gerados, permitindo que a especificação seja melhorada antes do programa ser escrito.

Nesse contexto, o TBM surge como uma abordagem aplicável para controlar a qualidade do software, assim como reduzir os custos associados ao processo de testes, visto que casos de teste podem ser gerados a partir da especificação do software, paralelamente ao seu desenvolvimento, utilizando procedimentos automáticos que podem ser menos suscetíveis a erros. Ele pode gerar melhorias nas atividades de teste por meio da simplificação do seu planejamento, da semi-automação de suas atividades, e controle, a partir da gerência dos testes e possibilidade de re-execução automática após modificações.

Segundo El-Far e Whittaker (2001), as vantagens de utilizar teste baseado em modelos são: comunicação entre desenvolvedores e testadores; geração automática de testes; atualização da suíte de testes. No que se refere à comunicação, uma vez que existe um modelo do comportamento da aplicação, então este pode ser utilizado como a base de comunicação entre testadores e desenvolvedores. Com o modelo do comportamento da aplicação, a geração de casos de teste pode ser facilmente automatizada. E quanto à atualização, uma vez alterado o modelo, facilmente pode ser feita a atualização da suíte de teste. Como desvantagem, os autores destacam a necessidade de conhecimento da notação do modelo. O testador deve estar familiarizado com a notação que será utilizada, o que culmina na demanda de tempo e investimento em treinamentos, além do tempo que

deve ser reservado para a obtenção do modelo. Outra desvantagem é a dependência da existência do modelo, uma vez que toda essa abordagem parte do modelo construído.

Utilizando o TBM a especificação de testes pode ser realizada informalmente através de linguagens naturais associadas ou não a tabelas de estados ou diagramas de transições. Essas especificações informais tendem a ser ambíguas e de difícil análise. Para conferir uma maior precisão e confiabilidade foram elaboradas as Técnicas de Descrição Formal (TDFs), que possuem um embasamento matemático que assegura a descrição de uma maneira precisa, a qual pode ser submetida à análise e à validação das propriedades requeridas.

De acordo com Mantovan (2006), uma especificação por meio de uma TDF deve apresentar as seguintes propriedades: ser clara, concisa e sem ambiguidades; ser completa, sem omitir nenhum detalhe; ser consistente; e, ser tratável, o que significa que a especificação pode ser submetida à análise para verificação e validação de suas propriedades.

Na literatura são encontradas diversas técnicas de modelagem, formais e semi-formais, que podem ser utilizadas com o Teste Baseado em Modelos, sendo elas: Máquina de Estados Finitos, *Statecharts*, Redes de Petri, SDL, Estelle e UML. Tais técnicas são descritas em ((Mantovan, 2006), (Cartaxo, 2006), (Fantinato, 2002)).

2.4.3 Processo de Teste

A concepção tradicional do processo de teste, como sendo uma fase final e independente do processo de desenvolvimento propriamente dito, tem se mostrado bastante ineficiente devido aos altos custos associados com a correção de erros encontrados e manutenção do software. Tal fato contribuiu para a definição de métodos e técnicas sistemáticas de teste que fazem do processo de teste, um conjunto de tarefas à parte que pode ser aplicado ao longo do processo de desenvolvimento (McGregor e Sykes, 2001).

Um processo de testes possui características próprias em relação ao processo de desenvolvimento dentro de um projeto de software. A aplicação de testes desde a concepção até a implantação do sistema viabiliza a identificação e correção de defeitos tão logo eles sejam percebidos, evitando assim, o acúmulo e o provável encadeamento dos mesmos. Uma especificação mal concebida pode originar um sistema com funcionalidades distintas das esperadas pelo usuário ou, até mesmo, ausentes. Uma análise mal elaborada pode resultar numa arquitetura que não permita atender determinados critérios de qualidade. Um projeto mal construído pode comprometer a evolução do software. Uma codificação mal feita pode ocasionar falhas de execução. Tais problemas revelam a necessidade de

disciplina na condução de testes de um software a fim de identificar, tão cedo quanto possível, os defeitos nele existentes.

Em (Crespo et al., 2004) é ressaltada a necessidade de se utilizar uma metodologia que possibilite a interação das atividades de teste a partir do início das especificações e que se estenda durante a construção do software até a entrega do sistema. Com isso, é possível obter melhorias drásticas na eficácia e eficiência dos testes e, consequentemente, aumentar a qualidade do software.

De acordo com McGregor e Sykes (2001), um processo de teste deve responder às seguintes questões:

- Quais partes do software serão testadas? Todos os componentes do software serão testados ou somente partes críticas e de alto risco?
- Como os testes serão realizados? Quais os critérios e técnicas para realização dos testes?
- Quem realizará os testes? Quem será responsável pelo planejamento dos testes? E pela execução dos testes?
- Quando os testes serão realizados? Em que momento do processo de desenvolvimento os testes serão executados?
- Onde os testes serão realizados? Em qual ambiente e qual a configuração do ambiente os testes ocorrerão?
- Qual a quantidade de testes adequada? Como decidir o que testar e quando finalizar os testes considerando os recursos limitados para a atividade?

O processo de teste deve basear-se em uma metodologia aderente ao processo de desenvolvimento, em pessoal técnico qualificado, em ambiente e ferramentas (McGregor e Sykes, 2001).

O aperfeiçoamento das atividades de teste, em termos de maior automação, melhor integração e aproveitamento das práticas de desenvolvimento de software, deve contribuir não apenas para a melhoria da qualidade dos produtos, mas também para propiciar condições que garantam que as organizações melhorem seus processos e os instanciem de forma correta.

2.4.4 Norma IEEE 829-1998

Independentemente do processo de software utilizado durante o desenvolvimento, existem artefatos diretamente relacionados à atividade de teste. A norma IEEE 829 [IEEE829-98] agrupa diversas soluções, apresenta algumas boas práticas na área e tem como proposta descrever um conjunto básico de documentos de teste de software. Na norma são definidos oito documentos, sendo eles (Crespo et al., 2004):

- Plano de Teste: apresenta o planejamento para execução do teste, incluindo a abrangência, abordagem, recursos e cronograma das atividades de teste. Identifica os itens e as funcionalidades a serem testadas, as tarefas a serem realizadas e os riscos associados com a atividade de teste;

- Especificação de Projeto de Teste: refina a abordagem apresentada no Plano de Teste e identifica as funcionalidades e características a serem testadas pelo projeto e por seus testes associados. Este documento também identifica os casos e os procedimentos de teste, se existirem, e apresenta os critérios de aprovação;

- Especificação de Caso de Teste: define os casos de teste, incluindo dados de entrada, resultados esperados, ações e condições gerais para a execução do teste;

- Especificação de Procedimento de Teste: especifica os passos para executar um conjunto de casos de teste;

- Diário de Teste: apresenta registros cronológicos dos detalhes relevantes relacionados à execução dos testes;

- Relatório de Incidente de Teste: registra qualquer evento que ocorra durante a atividade de teste e que requeira análise posterior;

- Relatório-Resumo de Teste: sumariza os resultados das atividades de teste associadas com uma ou mais especificações de projeto de teste e provê avaliações baseadas nesses resultados;

- Relatório de Encaminhamento de Item de Teste: identifica os itens encaminhados para teste no caso de equipes distintas serem responsáveis pelas tarefas de desenvolvimento e de teste.

2.4.5 Metodologia TOTEM (Testing of Object-Oriented Software SysTEms with the UML)

A metologia TOTEM tem por objetivo apoiar a derivação dos requisitos de teste funcionais a partir de artefatos UML. Os artefatos utilizados são: diagramas e descrições de casos de uso, diagramas de interação e diagramas de classe. Na Figura 2.4 é apresentado o diagrama de atividades da metologia TOTEM.

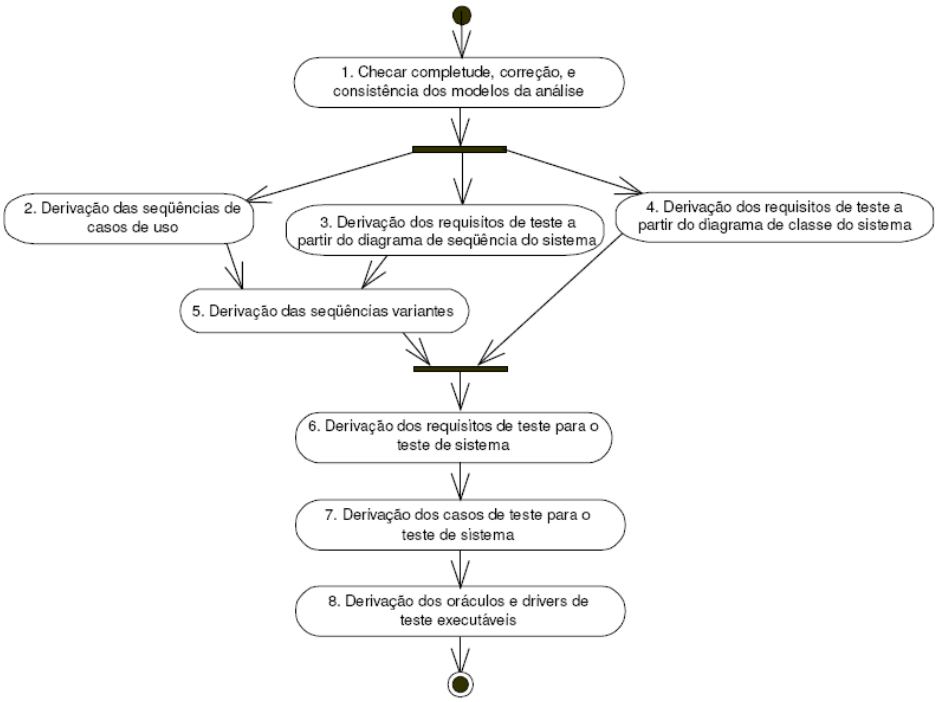

Figura 2.4: Diagrama de atividades dos passos da metologia TOTEM (Briand e Labiche, 2001).

No passo 1 os artefatos são verificados para garantir que estes são testáveis. Após essa verificação, os requisitos de teste são derivados a partir dos artefatos especificados, percorrendo os passos de 2 a 5. No passo 6 os requisitos são fundidos em um único conjunto de requisitos de teste, gerando o plano de teste. Por fim, são executados os passos 7 e 8 em que são derivados os casos de teste e os códigos para os oráculos. Para a geração dos requisitos de teste são utilizados os passos 2, 3 e 5, os quais são descritos como segue (Briand e Labiche, 2001):

- Derivação das sequências de casos de uso: os casos de uso possuem restrições e dependências em relação à sua execução, o que significa que eles devem ser executados em uma determinada ordem. A ordem de execução e as dependências

dos casos de uso podem ser visualizadas através do diagrama de atividades e da descrição estendida do caso de uso. Para obter a sequência dos casos de uso basta percorrer os caminhos do diagrama de atividades.

- Derivação dos requisitos de teste a partir do diagrama de sequência: O diagrama de sequência é reescrito como uma expressão regular e as condições para que um determinado caminho no diagrama de sequência seja habilitado são escritas em OCL (*Object Constraint Language*). Essas condições são obtidas a partir das condições associadas aos caminhos do diagrama de sequência. Em seguida, devem ser identificadas as sequências das operações que serão executadas

- Derivação das sequências variantes: a partir das sequências de casos de uso são geradas as sequências variantes. Uma sequência variante corresponde a um conjunto de condições que devem ser satisfeitas para que cada sequência do caso de uso possa ser testada.

2.4.6 Perfil UML2.0 de Teste

A linguagem UML se desenvolveu como foco principalmente na definição da estrutura de sistemas e de seu comportamento. No entanto, com a evolução de abordagens baseadas em modelos, por exemplo arquitetura dirigida a modelos (MDA), surgiu a necessidade de integração de atividades de teste de conformidade. Essa integração deveria permitir a especificação de modelos que capturassem as informações necessárias para realizar testes do tipo caixa-preta em implementações do sistema.

Para integrar a atividade de teste e modelos de sistema, foi desenvolvido pela OMG o Perfil UML 2.0 de Testes (U2TP - UML 2.0 *Testing Profile*), que define uma linguagem para projetar, visualizar, especificar, analisar, construir e documentar os artefatos de teste de sistemas (OMG, 2004).

O U2TP é uma linguagem de modelagem de testes que pode ser usada com tecnologias de componentes e linguagens orientadas a objeto, aplicadas em diversos domínios de aplicação. Em sua concepção foram utilizados dois princípios de projeto: Integração com UML e Reuso.

A integração com a UML refere-se ao fato de o perfil ser baseado no meta-modelo UML, desse modo usa os mesmos princípios e elementos dessa linguagem. O reuso está relacionado ao uso, onde possível, dos conceitos e elementos já presentes em UML.

O perfil foi projetado para apoiar o teste de implementações de sistema de maneira eficiente e tão automatizada quanto possível, tendo como foco o teste funcional caixa-preta.

O sistema sob teste (SUT) não é especificado como parte dos diagramas descritos usando U2TP: os elementos são importados a partir do modelo completo. O SUT é acessado a partir de suas interfaces públicas. No entanto, se o modelo de sistema define e oferece acesso a interfaces de subsistemas e componentes internos, esses também podem ser acessados em um teste: cada componente pode ser considerado um SUT menor, de escopo mais restrito, o que possibilita a descrição usando U2TP de testes em todos os níveis.

U2TP define apenas uma linguagem, e não um método. Deste modo, os usuários do perfil ficam livres para definir metodologias próprias, adaptadas às suas necessidades e domínios de aplicação.

O perfil é organizado em diferentes grupos de conceitos. Cada um desses grupos oferece elementos e relações a serem usadas na descrição de uma atividade de testes. A arquitetura do Perfil UML 2.0 de Testes encontra-se estruturada em quatro grupos de conceitos, são eles: Arquitetura de Teste, Comportamento de Teste, Dados de Teste e Temporização de Teste.

O grupo Arquitetura de Teste define os conceitos necessários para descrever os componentes usados em uma atividade de teste e o relacionamento entre eles. Centra-se nos aspectos estáticos usados durante a verificação de um sistema, como a identificação do sistema sob testes (SUT), que provê operações através de interfaces públicas acessadas por componentes de teste (*test components*), agrupados em contextos de teste (*test contexts*). Define também um árbitro (*arbiter*), que determina o veredito final do teste, e um escalonador (*scheduler*), que é responsável por controlar a execução. Os elementos UML deste grupo são representados na Figura 2.5 e descritos como segue:

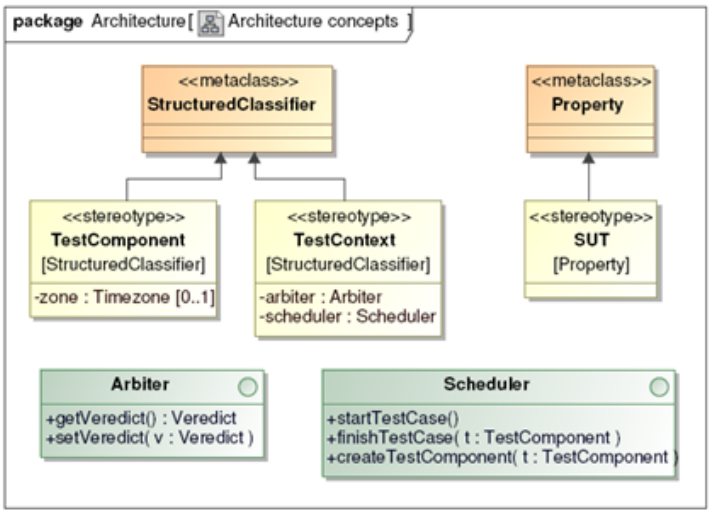

Figura 2.5: Arquitetura de Teste (OMG, 2004).

- *System under test* (SUT): O sistema sob testes é ativado através de um conjunto de interfaces públicas que expõe suas funcionalidades. Essas interfaces são importadas a partir do modelo completo do sistema. Como os componentes de teste se comunicam com o alvo apenas através delas, não existem outras maneiras de obter informação do SUT, caracterizando um teste tipo caixa-preta. Um SUT pode se apresentar através de diferentes níveis de abstração, como um sistema completo, um subsistema ou, ainda, um conjunto de componentes, como classes. A granularidade do SUT define o nível de teste (unitário ou de integração).

- *Test Component*: representa a implementação de um caso de teste, é normalmente uma classe que interage com o SUT ou com outros componentes. Ele executa uma sequência de ações na forma de estímulos, ativando funcionalidades do alvo, e de observações, recuperando informações. O componente pode realizar validações, verificando os dados recebidos do SUT e atualizando o veredito do teste, bem como registrar informações em *logs*.

- *Arbiter*: interface cuja implementação tem como propósito determinar o veredito final para o teste. Cada componente de teste informa a um árbitro central os seus vereditos locais. O árbitro central determina a política para o veredito final: por exemplo, indicar o teste inteiro como falho se uma certa percentagem dos casos de teste não terminar com sucesso.

- *Scheduler*: interface cuja implementação controla a execução dos diferentes componentes de teste. Ele realiza a criação e destruição dos componentes, inicia sua execução e interage com o árbitro para informar o momento de calcular o veredito final do teste.

- *Test Context*: estrutura usada para agrupar componentes de teste. Ela é uma classe associada a um conjunto de componentes de teste, a uma instância de árbitro, uma instância do SUT e uma instância de um escalonador. Sua implementação pode ser tanto código executável como *scripts*. Este elemento foi projetado para ser gerado automaticamente por ferramentas, usando as descrições dos demais elementos.

O segundo grupo, Comportamento de Teste, define os conceitos relacionados aos aspectos dinâmicos dos procedimentos, tais como: invocação de ações, determinação de vereditos e registros (*logs*). Neste grupo há, também, elementos que associam casos de uso a casos de teste. A Figura 2.6 ilustra os elementos deste grupo, os quais são definidos como segue:

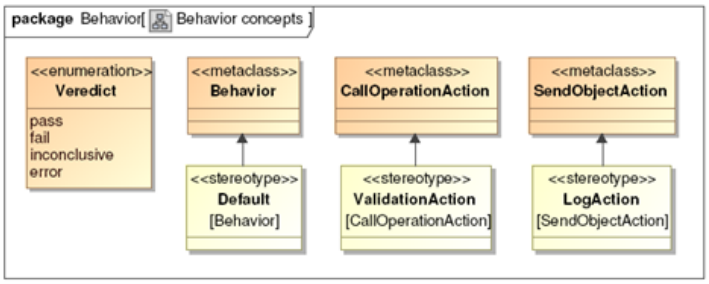

Figura 2.6: Comportamento de teste (OMG, 2004).

- *Veredict*: enumeração que contém ao menos os valores *fail*, *inconclusive*, *pass* e *error*, que indicam o resultado da execução de um teste. O valor *pass* indica que o SUT se comportou como o previsto. *Fail*, por outro lado, indica que o comportamento do SUT não foi o esperado. Se a execução do teste não pôde determinar um resultado, o valor *inconclusive* é usado. O veredito *error* é usado para indicar que o sistema de teste teve problemas.

- *Validation Action*: ação executada durante um teste que informa um resultado local ao árbitro central.

- *Default*: artifício usado para simplificar os modelos de teste, permitindo construir definições parciais de componentes de teste de maneira compacta. Uma especificação *default*, que pode ser apresentada como diagramas de sequência ou de estado, indica qual ação o sistema de teste deve tomar quando o SUT não se comporta da maneira esperada. Como essas especificações podem ser referenciadas por diferentes componentes, os modelos se tornam mais compactos.

- *Test Log*: componente através do qual os contextos de teste podem criar rastros de execução e indicar informações de interesse. Esses registros tornam-se parte da especificação do sistema de teste.

O terceiro grupo de conceitos é o de Dados de Teste, que define a sintaxe e semântica dos dados usados como entrada e saída dos procedimentos de teste. Dados explícitos e classes de equivalência formam conjuntos de dados (*data sets*); essas classes de equivalência especificam regras para formação de dados de entrada ou saída. Valores coringas (*wild-cards*), em adição aos já presentes em UML padrão, são oferecidos. Os elementos seletores de dados (*data selectors*) são usados para facilitar a criação de estratégias de teste. Os elementos que compõe este grupo são mostrados na Figura 2.7 e uma breve descrição é apresentada:

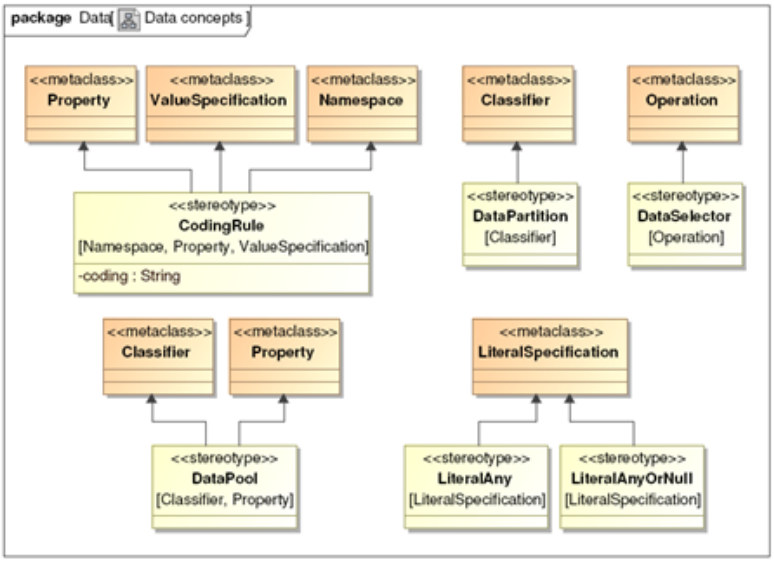

Figura 2.7: Dados de teste (OMG, 2004).

- *Wildcards*: valores literais usados para especificar de forma relaxada um dado fornecido a um SUT ou recebido dele durante um teste.
- *Data Partition*: estabelece uma classe de equivalência para valores a serem fornecidos a um SUT ou recebidos durante um teste. É usada como uma forma de diferenciar, visivelmente, os dados, atribuindo a eles nomes como, por exemplo, NumerosCPFValidos.
- *Data Pool*: mecanismo usado para associar conjuntos de dados a contextos de teste. É uma classe que contém partições de dados ou valores explícitos e uma associação a um contexto de teste.
- *Data Selector*: operações que definem as estratégias usadas pelo sistema de teste para fornecer e verificar os dados de entrada e saída do SUT. São associados a uma partição ou repositório de dados.
- *Coding Rules*: strings usadas para referenciar padrões de codificação e de protocolos de transmissão de dados externos ao perfil de teste, como CORBA ou XML.

O grupo Temporização de Teste apresenta os conceitos de tempo usados para sincronização entre sistemas e para verificação de desempenho. Temporizadores (*timers*) são definidos, permitindo especificar limites de tempo para operações em um caso de teste. O conceito de fuso horário (*timezones*) agrupa componentes de teste sob uma mesma percepção de tempo. Na Figura 2.8 são representados os elementos UML deste grupo.

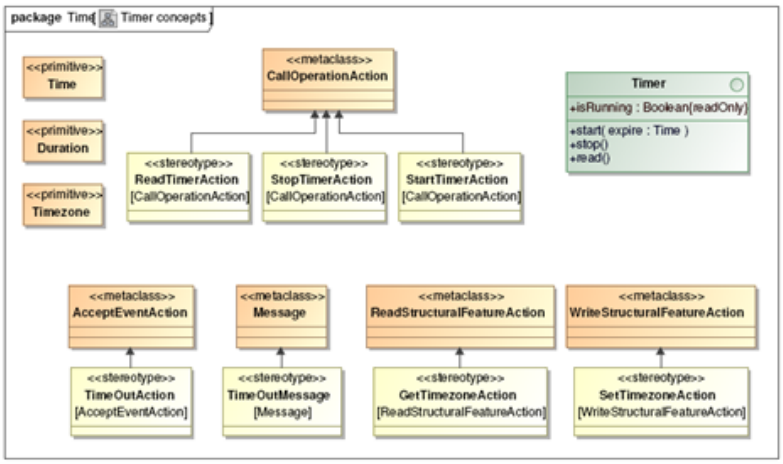

Figura 2.8: Temporização de teste (OMG, 2004).

O Perfil UML 2.0 de Testes possibilita projetar e descrever atividades de verificação e validação de sistemas em uma linguagem padronizada, desenvolvida com foco em abordagens MDA e em automação.

2.5 Engenharia Experimental

No contexto da engenharia de software, os estudos empíricos têm adquirido importância ao longo do tempo. Eles podem ser definidos com uma forma de se obter informações sobre um objeto de pesquisa. Desse modo, auxiliam na construçãao de conhecimento com base em observações e evidencias empíricas. Parte da comunidade de pesquisa está voltando suas atenções para um conjunto de técnicas agregadas sobre a denominação de Engenharia de Software Experimental (Basili, 2006).

A Engenharia de Software Experimental tem como objetivo apontar indícios sobre hipóteses levantadas a respeito de determinados artefatos gerados durante o desenvolvimento de software.

Um experimento é um tipo de estudo empírico realizado em um ambiente controlado, passível de observação e onde as hipóteses podem ser analisadas. Essa forma de estudo está focada em variáveis específicas com um potencial estatístico significativo, sendo a forma mais efetiva para provisão de evidência empírica e refinamento de conhecimento. Os resultados obtidos são analisados para identificar variáveis chaves e o relacionamento entre elas. Essa análise pode ser apoiada por habilidade humana, experiências, conhecimento de domínio entre outros.

A realização de experimentos tem por objetivo avaliar, caracterizar, prever, controlar e melhorar produtos, processos, recursos, modelos, teorias entre outros. Além disso, presume um relacionamento de causa e efeito . Um experimento é desdobrado em diversas atividades, as quais possuem complexidade variável.

Segundo (Wohlin et al., 2000), a experimentação caracteriza-se por um processo que tem um início a partir da Definição do experimento passando pelo Planejamento, Execução, Análise e Interpretação, Apresentação e Empacotamento.

Na fase de **Definição**, o experimento é definido em termos dos objetivos e metas do estudo. Desse modo, devem ser respondidas questões relacionadas ao objeto do estudo (o que será estudado), propósito (intenção do estudo), foco de qualidade (aspecto de qualidade a ser estudado), perspectiva (ponto de vista em que os resultados serão interpretados) e contexto (ambiente no qual o experimento será realizado).

A etapa **Planejamento** prepara como o estudo experimental será conduzido, através da definição dos seguintes elementos: contexto, hipóteses, variáveis, participantes e projeto experimental, contendo instrumentação e avaliação da validade.

Após o planejamento do estudo, tem-se a **Execução** do mesmo com o propósito de se coletar os dados para serem analisados. Essa etapa deve haver uma monitoração por parte do pesquisador para ter certeza que tudo seja feito de acordo com o planejado, qualquer viés nessa etapa pode afetar a validade do estudo.

Na fase de **Análise e Interpretação** as hipóteses elaboradas são comparadas com os dados obtidos através de ferramentas estatísticas adequadas. Os resultados gerados nesta fase são usados pelo experimentador na tentativa de rejeitar a hipótese nula e confirmar a hipótese alternativa.

A etapa de **Apresentação e Empacotamento** os resultados obtidos com o experimento são apresentados e empacotados, através de documentatação (artigos de pesquisa), ou ainda compondo uma base de conhecimento.

No planejamento de experimentos é importante considerar a necessidade de replicação ou complementação do estudo a fim de ampliar a base de dados, a credibilidade e o índice de confiabilidade. A fase de apresentação e empacotamento oferece suporte para a replicação, oportunidades de variação e combinação de resultados.

2.6 Considerações Finais

Os conceitos e técnicas apresentados neste capítulo oferecem o embasamento necessário para a contextualização, fundamentação e condução de estudos experimentais deste trabalho.

O Capítulo 3 discute os trabalhos relacionados e sua relevância para o desenvolvimento da Abordagem Integrada de Desenvolvimento e Teste de Software para Equipes Distribuídas.

Capítulo

3

Trabalhos Relacionados

Este capítulo apresenta os processos de desenvolvimento e teste de software que nortearam o desenvolvimento deste trabalho e uma análise comparativa entre os mesmos.

3.1 Processos de Desenvolvimento de Software

Esta seção apresenta, sucintamente, os processos de desenvolvimento, identificando suas atividades, artefatos e papéis, sendo eles: RUP (seção 3.1.1), *eXtreme Programming* (seção 3.1.2), *Agile Unified Process* (seção 3.1.3), *Extended Workbench Model* (seção 3.1.4) e *Local Agile Game-based Process* (seção 3.1.5).

3.1.1 Rational Unified Process (RUP)

O RUP é um processo de desenvolvimento de software iterativo, incremental e adaptável, podendo ser customizado para diversos tipos e tamanhos de produtos e projetos de software. Identifica cada ciclo de desenvolvimento do projeto em quatro fases, cada uma com marcos de finalização definidos, os chamados *milestones* (Kruchten, 2003).

Os *milestones* indicam o progresso do projeto e são usados como base para decisões para continuar, cancelar, ou mudar o rumo do projeto. O RUP apresenta quatro fases: **início**, que tem por objetivo determinar o escopo do desenvolvimento; **elaboração**, que refere-se ao planejamento das atividades e recursos necessário; **construção**, relacionada à implementação do software; e **transição**, que tem por objetivo disponibilizar o produto.

O RUP estrutura o projeto fazendo uma divisão bem definida de atividades (tarefas e papéis) e apresenta uma coleção de artefatos, nos quais a comunicação se baseia. O processo é dividido em nove disciplinas: Gerência de Projetos, Modelagem de Negócios, Requisitos, Análise e Projeto, Implementação, Teste, Instalação, Configuração e Controle de Mudanças e Ambiente.

A disciplina de Gerência de Projetos, desempenhada pelo Gerente de Configuração e Engenheiro de Processo, oferece suporte para equilibrar as metas em termos de custos e prazos e gerenciar o risco para a entrega de um produto que satisfaça as necessidades do cliente. A Modelagem de Negócio visa entender a estrutura e a dinâmica da organização na qual o software será inserido e envolve os papéis de Analista de Processos de Negócios, Projetista de Negócio e Revisor de Negócios.

Na disciplina de Requisitos as necessidades do sistema são traduzidas na forma de casos de uso, e os papéis envolvidos são: Analista de Sistema, Especificador de Caso de Uso e Projetista de Interface do Usuário. A Análise e Projeto é responsável pela especificação da forma de implementação dos requisitos, sendo desempenhada pelos papéis de Arquiteto e Projetista.

A implementação das classes e objetos na forma de componentes ocorre na disciplina Implementação, que é realizada pelos papéis de Implementador, Arquiteto e Revisor de Código. A disciplina de Teste, desempenhada pelos papéis Projetista de Teste e Testador, é responsável por testar e verificar se o produto funciona como o esperado, documentando falhas e problemas.

A disciplina Configuração e Controle de Mudanças envolve o Gerente de Configuração e o Gerente de Controle de Mudanças e concentra-se na garantia da rastreabilidade das versões de um projeto. O suporte adequado à organização do projeto em ferramentas, métodos e processos é provido pela disciplina Ambiente, que conta com os papéis de Engenheiro de Processo, Escritor Técnico, Analista de Sistema, Analista do Processo Empresarial, Projetista de Interface do Usuário, Projetista de Teste, Arquiteto, Especialista de Ferramenta e Administrador de Sistema. Por fim, tem-se a disciplina de Distribuição que tem por objetivo distribuir, instalar, testar em campo e treinar os usuários. Os papéis envolvidos são o Gerente de Distribuição, Gerente de Projeto, Escritor Técnico, Desenvolvedor de Curso, Artista Gráfico, Testador e Implementador.

3.1.2 eXtreme Programming (XP)

O XP é uma metodologia de desenvolvimento ágil, com iterações curtas, que objetiva tornar o projeto flexível, diminuindo o custo a possíveis mudanças e utiliza o código

gerado como um indicador de progresso do projeto. A metodologia XP é baseada em quatro valores: comunicação, simplicidade, *feedback* e coragem.

A **comunicação** é o valor mais importante, pois a maioria dos problemas que ocorrem nos projetos tem sua causa associada a falhas na comunicação. Utilizando o XP espera-se que a comunicação seja qualitativamente alta e freqüente. Os clientes comunicam requisitos funcionais através de estorietas escritas que são usados para guiar o desenvolvimento. A comunicação entre os desenvolvedores ocorre sempre que necessário e, diariamente, são realizadas reuniões que alinham a equipe ao andamento global do projeto.

O valor **simplicidade** refere-se ao fato de que os desenvolvedores devem objetivar a construção do sistema mais simples possível, que realize de forma eficaz o trabalho a que se propõe. E, conforme novos requisitos são adicionados no sistema, este é modificado o quanto for necessário, mas o mínimo possível, implementando apenas o que é necessário e realmente importa ser construído.

O ***feedback*** no XP ocorre em diversos aspectos na organização do trabalho. A integração constante do sistema fornece aos desenvolvedores informação imediata sobre o estado do sistema e cada execução de testes oferece aos desenvolvedores *feedback* imediato em relação a possíveis inconsistências no código. Obter o *feedback* mais cedo significa ter mais tempo disponível para reagir.

O valor **coragem** passa pela mudança de paradigma necessária para adoção de uma metodologia ágil, mas atinge também o processo em todos os níveis e tarefas. Significa tomar as decisões na hora em que elas precisam ser tomadas. Seguindo esse valor, se uma funcionalidade não está funcionando, conserte-a. Se algum código não parece estar bem escrito, refatore-o.

O ciclo da metodologia XP é composto pelas seguintes etapas:

1. Exploração: nessa etapa o cliente escreve cartões de estórias, cada um contendo uma funcionalidade desejada para o primeiro *release*.

2. Planejamento: definição de prioridades entre as estórias junto com o cliente. Nesta etapa é realizada a estimativa de esforço e o cronograma para cada uma das estórias elaboradas anteriormente.

3. Iterações para *Release*: são realizadas diversas iterações até o primeiro *release* ser completado. A primeira iteração consiste na criação do sistema com toda a arquitetura, nas iterações seguintes são adicionadas às funcionalidades de acordo com as prioridades estabelecidas.

4. Validação para Produção: esta etapa compreende a realização de testes extensivos e verificações para validação do software.

5. Manutenção: após o primeiro *release* para produção, há novas edições do sistema com novas funcionalidades.

6. Morte: quando não há mais estórias a serem implementadas e o cliente está satisfeito com o código.

3.1.3 Agile Unified Process (AUP)

O AUP é uma versão simplificada do RUP, que utiliza técnicas e conceitos de métodos ágeis para descrever de uma maneira simples e fácil de compreender o desenvolvimento de software.

No AUP são realizadas sete disciplinas: Modelo, Implementação, Teste, Implantação, Gerenciamento de Configuração e Gerenciamento de Projetos. A disciplina Modelo compreende o ramo em que se insere a organização, o problema para o qual o software está sendo desenvolvido e a identificação da solução para resolvê-lo. Na disciplina Implementação os modelos são tranformados em códigos executáveis.

O Teste é uma disciplina em que é realizada uma avaliação objetiva com o intuito de assegurar a qualidade, encontrar defeitos, validar se o sistema faz o que se propõe e verificar se os requisitos são atendidos. A Implantação é responsável pela entrega do sistema.

Na disciplina Gerenciamento de Configuração é realizado o gerenciamento do acesso a todos os artefatos do projeto, incluindo o rastreamento das várias versões, controle e gerência das alterações. A disciplina Gerenciamento de Projetos é responsável por direcionar as atividades que ocorrem no projeto, por exemplo: gestão de risco e gestão de recursos humanos. E, Ambiente é a disciplina que objetiva assegurar que os processos, a direção do projeto e as ferramentas utilizadas estão acessíveis a toda a equipe de desenvolvimento.

3.1.4 Extended Workbench Model

O *Extended Workbench Model* tem sido amplamente utilizado na Siemens. Nessa abordagem há uma pequena equipe central que realiza o planejamento do projeto, defininindo os papéis e responsabilidades dos times remotos, e as atividades no início de cada fase, por exemplo, análise de requisitos, projeto de arquitetura e plano de projeto. As equipes distribuídas são responsáveis pelas atividades de desenvolvimento e teste (Paulish, 2007).

Segundo Avritzer et al. (2007) no *Extended Workbench Model* tem-se como práticas gerenciamento centralizado, desenvolvimento iterativo, diminuição da comunicação entre times e especificação formal dos requisitos. Com isso, os requisitos do sistema, arquitetura e testes do sistema são executados de modo centralizado, utilizando ciclos de iterações curtos (cerca de 15 dias), a comunicação entre os times é gerenciada pela equipe central e, com o intuito de minimizar a comunicação entre os times remotos realiza-se a especificação dos requisitos utilizando métodos formais.

O time central é responsável por desempenhar os papéis de Engenheiro de Requisitos, Arquiteto, Engenheiro de Integração, Gerente de Infraestrutura e Administrador da ferramenta de colaboração. As equipes remotas desempenham os papéis de Desenvolvedores, Líderes de Testes, Engenheiros de Requisitos, Arquiteto de Sistemas, Gerente de Qualidade, Gerente de Infraestrutura e Engenheiro de Processo e Documentação (Urdangarim, 2008).

3.1.5 Local Agile Game-based Process (LAGPRO)

O LAGPRO é um processo para ser utilizado por equipes locais para reforçar a coordenação e motivação em um contexto de desenvolvimento distribuído de software. É um processo iterativo, em que cada iteração é dividida em fases, as quais têm duração de duas semanas.

A estrutura e os princípios gerais do LAGPRO são baseados no AUP, simplificando suas disciplinas e fases. A disciplina *Model*, foi esboçada utilizando a construção de modelos simplificados no início de cada nova iteração e de modelos de revisões ao final de cada iteração. A metodologia *Test-Driven Development* (TDD) foi utilizada na disciplina implementação, onde os times de testes definiam os testes unitários automatizados e os times de desenvolvimento implementavam as regras de negócio (Ribeiro et al., 2006).

O LAGPRO buscou introduzir motivação em equipes utilizando elementos de jogos e competição. Segundo Ribeiro et al. (2006), as tentativas de execução do processo falharam devido à rejeição por parte dos membros do time com relação à característica competitiva.

3.2 Análise Comparativa dos Processos de Desenvolvimento

Os processos RUP e XP foram escolhidos por serem amplamente utilizados, quer na academia como na indústria, sendo uma metodologia tradicional e uma ágil, respectiva-

mente. Em (Rocha et al., 2008) é apresentado um relato de experiência sobre a adaptação do RUP para pequenas equipes de desenvolvimento distribuído. O AUP foi selecionado por ser uma simplificação do RUP que utiliza conceitos de métodos ágeis. E os demais, o *Extended Workbench Model* e o LAGPRO, por serem utilizados no contexto de DDS, conforme pode ser visto em (Ribeiro et al., 2006), (Paulish, 2007) e (Urdangarim, 2008).

A seguir será apresentada a comparação entre os processos de desenvolvimento de software mencionados acima. Para tanto, foi identificado um conjunto de critérios de comparação baseando-se nas características de desenvolvimento distribuído de software identificadas em (Urdangarim, 2008) (critérios de A - G), nos elementos básicos de um processo de software (critérios de H - J) e no tipo de metodologia (critério K e L). Os critérios a serem utilizados na comparação visando identificar qual dos processos melhor atende às necessidades dos projetos DDS, são:

- (A) Diversidade: Identifica se o processo analisado oferece mecanismos para controle, coordenação e comunicação das diferenças (cultural e idioma) entre as equipes envolvidas no projeto.
- (B) Requisitos: Caracteriza se o processo analisado trata a especificação formal dos requisitos, que tem por objetivo amenizar os problemas de ambiguidades nos artefatos e com isso, minimizar a comunicação.
- (C) Centralizado: Identifica se o processo analisado apresenta uma centralização do controle das atividades do projeto.
- (D) Descentralizado: Identifica se o processo analisado é caracterizado pela descentralização do controle das atividades do projeto.
- (E) Acompanhamento: Identifica se o processo caracteriza-se por acompanhar, periodicamente, a evolução dos trabalhos dos times remotos.
- (F) Colaboração: Caracteriza se o processo permite o uso de ferramentas de colaboração entre os times como meio de compartilhamento de informações.
- (G) Confiança: Identifica se o processo é capaz de promover confiança entre os membros dos diferentes times. A confiança pode ser alcançada oferecendo visibilidade adequada do projeto, por meio da divulgação das informações e por meio de mecanismos de percepção.
- (H) Papéis: Identifica se o processo possui uma organização rígida na definição dos papéis a serem desempenhados.

- (I) Artefatos: Identifica se o processo apresenta a definição dos artefatos requeridos e gerados em cada fase.
- (J) Atividades: Identifica se o processo define as atividades que devem ser realizadas em cada fase.
- (K) Metodologia tradicional: Caracteriza se o processo apresenta o enfoque de uma metodologia tradicional de desenvolvimento de software.
- (L) Metodologia ágil: Caracteriza se o processo apresenta o enfoque de metodologia ágil.

Segundo Urdangarim (2008), definir os critérios de análise é uma atividade que requer um considerável esforço intelectual e que, geralmente, dá margem para questionamentos que podem ir além do escopo do trabalho em questão. Desse modo, com os critérios expostos não se espera esgotar as perspectivas de comparação dos processos.

Na Tabela 3.1 os processos selecionados e descritos na Seção 3.1 são comparados em relação aos critérios mencionados acima, cuja análise encontra-se na sequência.

Tabela 3.1: Análise Comparativa dos Processos

Processo/Critérios	A	B	C	D	E	F	G	H	I	J	K	L
RUP				x				x	x	x	x	
XP				x	x	x	x	x	x	x		x
AUP				x				x	x	x		x
Extended Workbench Model		x	x	x	x			x	x			x
LAGPRO		x	x	x	x	x	x	x	x			x

Analisando a Tabela 3.1 acima, é possível observar que nenhum dos processos atende todas as necessidades dos projetos desenvolvidos com equipes geograficamente distribuídas.

Não há um processo que contemple os critérios relacionados ao DDS (critérios de A - G) e apresente os elementos básicos (papéis, artefato e atividades). Desse modo, percebe-se a inexistência de um processo sistemático que contemple as características dessa nova configuração de desenvolvimento. Para tanto, devem ser definidos as atividades, os artefatos e os papéis necessários em cada etapa do desenvolvimento utilizando a abordagem distribuída.

O RUP, o XP e o AUP definem o conjunto mínimo de elementos que compõem um processo. Entretanto, não apresentam procedimentos para tratar as questões relacionadas aos problemas de DDS, que são representadas pelos critérios de A a G.

O *Extended Workbench* e o LAGPRO, embora tenham sido definidos para serem utilizados por projetos que são desenvolvidos de modo distribuído, não possuem mecanismos

para explicitar a diversidade, tratam apenas a especificação formal dos requisitos. No *Extended Wokbench*, também, não são contemplados os fatores relacionados à colaboração e confiança. Além disso, não foram encontrados relatos sobre os artefatos gerados. E, no caso do LAGPRO, também, não foram identificadas as atividades.

Desse modo, é possível evidenciar a necessidade de um processo que atenda essas novas demandas geradas pela abordagem distribuída e apresente a definição dos elementos que um processo deve contemplar.

O processo de desenvolvimento de software tem passado por intensas transformações nos últimos anos. E nesse contexto, novas abordagens de desenvolvimento têm surgido objetivando que muitos dos problemas encontrados nos projetos anteriores não sejam repetidos. Dentre os problemas mais comuns aos projetos de software podem ser destacados: altos custos para evolução, inconsistência entre documentação e sistema final, pouca ou quase que total falta de portabilidade e baixa confiabilidade. É sabido que o sucesso de um projeto de desenvolvimento de software inicia-se no devido planejamento e na escolha de uma metodologia compatível com as características do mesmo, o que reforça a necessidade de um processo de desenvolvimento que contemple de forma efetiva as necessidades do DDS.

Audy e Prikladnicki (2008), destacam que em um ambiente de desenvolvimento distribuído, é fundamental um processo de desenvolvimento comum à equipe pois, uma metodologia auxilia diretamente na sincronização, fornecendo a todos os membros da equipe uma nomenclatura comum de tarefas e atividades, e um conjunto comum de expectativas aos elementos envolvidos no processo.

Segundo Rocha et al. (2008), quando o contexto é Desenvolvimento Distribuído, o cenário muda com relação ao desenvolvimento de software tradicional, pois as variáveis e os riscos aumentam. Então, se não houver uma metodologia adequada para o processo de desenvolvimento, o projeto terá boas chances de não corresponder ao planejamento inicial. Em sua primeira percepção sobre o uso e estudo de DDS, Rocha et al. (2008) ressaltam a necessidade de processos mais adequados, pois, com base nas pesquisas realizadas, não foi fácil a identificação de modelos de referência de processos para o ambiente DDS.

3.3 Processos de Teste de Software

Nesta seção são descritos alguns processos de teste, identificando suas atividades, artefatos e papéis, sendo eles: RUP (seção 3.3.1), SiTest (seção 3.3.2) e Processo do CenPRA (seção 3.3.3).

3.3.1 Teste no RUP

No RUP o fluxo de teste é desempenhado por três papéis: Projetista, Implementador e Testador; e, composto por cinco atividades: Planejar, Projetar, Implementar, Executar Teste de Integração, Executar Teste de Sistema e Avaliar (Kruchten, 2003).

A atividade Planejar, realizada pelo Projetista, identifica e descreve o teste que será implementado e executado, e produz como artefato o plano de teste que contém informações sobre o propósito e as metas do teste e identifica as estratégias e recursos a serem utilizadas nas etapas subsequentes. Na atividade Projetar, o Projetista descreve e gera o modelo de teste que inclui os casos e procedimentos de teste.

A atividade Implementar os procedimentos de teste são implementados pelo Implementador produzindo o *script* de teste. Nas atividades Executar Teste de Integração e Executar Teste de Sistema, o Testador deve assegurar que os componentes do sistema colaboram e funcionam como planejado, respectivamente. Durante a execução dos testes os dados são capturados e o artefato Resultados de Teste é produzido.

A atividade Avaliar é realizada pelo Projetista e tem como objetivo entregar e avaliar as medidas quantificáveis de teste para determinar a qualidade do sistema em teste e do processo de teste. Nesta atividade é gerado um documento contendo um resumo dos testes.

3.3.2 SiTest

A metodologia foi desenvolvida com base no modelo de qualidade CMMI (*Capability Maturity Model Integration*) e encontra-se estruturada em cinco fases, as quais apresentam atividades e artefatos (Silva et al., 2006).

Na fase inicial, a de Planejamento, são realizadas as seguintes atividades: levantamento dos requisitos necessários para a realização dos testes; selecionar os módulos a serem testados; levantamento dos riscos e ponssibilidades de mitigação; identificação dos indicadores de desempenho; definição da equipe de teste bem como suas competências e atividades; e elaboração do cronograma inicial. Como artefatos desta fase são produzidos o plano de teste e o cronograma preliminar.

A etapa subsequente, Preparação, é composta pelas atividades: determinação da sequência dos componentes para integração; levantamento da configuração do ambiente para os testes e controle das mudanças; estabelecimento do ambiente de testes de integração, verificação e validação; instalação e configuração das ferramentas de teste; e, elaboração dos projetos de testes. Nesta fase são gerados três artefatos: o projeto de testes; relatório de equipe e suas respectivas atividades; e, ferramentas de testes disponíveis.

Na fase de Especificação são definidas as atividades da equipe de testes, os dados de entrada e saída são determinados e os casos e roteiros de testes são elaborados. Como saídas dessa fase tem-se a especificação dos roteiros e casos de testes. Após a Especificação tem-se a fase de Execução a qual é composta pelas seguintes atividades: execução dos testes de unidade, integração, sistemas e aceitação; elaboração da documentação formal de erros; análise dos resultados; e, solicitação das correções e alterações. Durante a Execução são produzidos o registro de teste e registro de consolidação de testes, registro de defeitos conhecidos, diários de testes, relatório de conformidades e não-conformidades e o relatório de incidente de testes.

A última fase da metodologia é a Entrega. Essa etapa consiste nas seguintes atividades: elaboração do relatório final de teste; elaboração de um resumo sobre o processo de teste e sua aplicação no projeto; e, registro da base de conhecimento do processo. Como saídas dessa fase são gerados os registros de testes, o resumo de testes e relatório final de testes.

3.3.3 Processo CenPRA

O Processo de teste do CenPRA é baseado em artefatos de teste definidos na norma IEEE Std 829-1998 e têm as suas atividades definidas e ordenadas de forma a permitir que o teste seja eficiente e eficaz. O Processo de teste do CenPRA contém subprocessos de teste e artefatos associados. Em cada subprocesso os papéis envolvidos são: Gerente, Executor, Cliente e Fornecedores. A seguir os subprocessos: Planejar, Projetar, Executar, e Registrar são descritos (Crespo et al., 2004).

O subprocesso Planejar envolve a realização das seguintes atividades: definir contexto do teste, descrevendo resumidamente os itens e características a serem testadas; caracterizar os itens de teste; identificar funcionalidades e características que devem ser testadas para cada item de teste; estabelecer abordagens e critérios para medir a abrangência dos testes, determinar o término, aprovação ou reprovação dos itens de teste, suspensão e retomada dos testes e de rastreabilidade; definir os produtos a serem gerados; definir atividades de teste; estabelecer requisitos de ambiente; estabelecer responsabilidades; estabelecer equipe e treinamento necessários; elaborar cronograma; identificar riscos e estabelecer contingências. O artefato produzido por este subprocesso é o Plano de Teste.

No subprocesso Projetar a abordagem de teste definida no planejamento é refinada. As atividades que compõe esse subprocesso são: detalhar as abordagens e critérios; especificar casos de teste estabelecendo as entradas requeridas para sua execução; estabelecer requisitos de ambiente de teste, tais como hardware, software e recursos humanos; definir objetivo dos procedimentos de teste; e definir os passos dos procedimentos de teste. As

saídas desse subprocesso são a Especificação de Projeto de Teste, Especificação de Casos de Teste e Especificação de Procedimento de Teste.

O terceiro subprocesso é o Executar que corresponde à realização dos passos do Procedimento de Teste. Este subprocesso é formado pelas seguintes atividades: executar a sequência de ações prévias necessárias para a execução do teste; Executar as ações necessárias para iniciar a execução do teste; registrar os resultados da execução e das medições do teste; e, executar as ações para suspender, retomar, parar e encerrar os testes.

O subprocesso Registrar tem como objetivos documentar cronologicamente os detalhes relevantes relacionados com a execução dos testes, registrar os eventos que ocorrem durante a execução e requerem análise e, descrever resumidamente os resultados e avaliações. Para tanto são realizadas as seguintes atividades: descrição do teste; registro das atividades e eventos; descrição dos incidentes de teste; determinação do impacto do incidente; descrição resumida dos itens de teste; descrição de desvios das especificações; descrição da abrangência; descrição resumida dos resultados; avaliação dos itens de teste e descrição resumida da atividade de teste. Os artefatos produzidos são: Diário de Teste, contendo a descrição do teste e registro de atividades e eventos; Relatório de Incidentes com o resumo e descrição do impacto do incidente; e, Relatório-Resumo de Teste constituído pelo Resumo dos Itens de Teste, Desvios das Especificações, Avaliação da Abrangência do Teste, Resumo de Resultados, Avaliação dos Itens de Teste e o Resumo de Atividades.

3.4 Análise Comparativa dos Processos de Teste

Na literatura não foram encontrados, até o momento da elaboração dessa dissertação, relatos de processos de teste utilizados no Desenvolvimento Distribuído de Software. Como há relatos de uso do RUP, XP e AUP, é importante analisar como a fase de testes é vista nesses processos. A SiTEST foi selecionada por ser uma metodologia de teste que apoia o ciclo de vida do software e ser baseada no modelo de qualidade CMMI. E o processo do CenPRA por ser amplamente utilizado (Crespo et al., 2004).

Os critérios utilizados na comparação dos processos de teste são:

- (A) Casos de Teste: Caracteriza se o processo analisado trata a especificação formal dos casos de teste.
- (B) Centralizado: Identifica se o processo analisado apresenta uma centralização do controle das atividades de teste.

- (C) Descentralizado: Identifica se o processo analisado é caracterizado pela descentralização do controle das atividades de teste.
- (D) Acompanhamento: Identifica se o processo caracteriza-se por acompanhar periodicamente a evolução dos trabalhos dos times remotos.
- (E) Colaboração: Caracteriza se o processo permite o uso de ferramentas de colaboração entre os times como meio de compartilhamento de informações.
- (F) Confiança: Identifica se o processo é capaz de promover confiança entre os membros dos diferentes times. A confiança pode ser alcançada oferecendo visibilidade adequada do projeto, por meio da divulgação das informações e por meio de mecanismos de percepção.
- (G) Papéis: Identifica se o processo possui uma organização sistemática na definição dos papéis a serem desempenhados.
- (H) Artefatos: Identifica se o processo apresenta a definição dos artefatos requeridos e gerados em cada fase.
- (I) Atividades: Identifica se o processo define as atividades que devem ser realizadas em cada fase.
- (J) Metodologia tradicional: Caracteriza se o processo apresenta o enfoque de uma metodologia tradicional de desenvolvimento de software.
- (K) Metodologia ágil: Caracteriza se o processo apresenta o enfoque de metodologia ágil.

Na Tabela 3.2 é apresentada a comparação entre os processos de teste.

Tabela 3.2: Análise Comparativa dos Processos de Teste

Processo/Critérios	A	B	C	D	E	F	G	H	I	J	K
RUP	x	x					x	x	x	x	
SiTest		x					x	x	x	x	
CenPRA		x					x	x	x	x	
XP		x					x	x	x		x
AUP		x					x	x	x		x

No RUP são especificados os papéis envolvidos, atividades e artefatos gerados na Disciplina de Teste. No entanto, não retrata a especificação formal dos casos de teste

que é um fator importante quando o desenvolvimento é realizado de forma distribuída por amenizar os problemas de comunicação decorrentes da ambiguidade e imprecisão dos artefatos.

Na metodologia SiTest e no processo do CenPRA os testes são realizadas ao longo do ciclo de vida do software. Tanto SiTest quanto o processo do CenPRA definem atividades, papéis e artefatos gerados em cada etapa. Entretanto, não apresentam mecanismos para tratar a especificação formal dos casos de teste. Mesmo não sendo encontrados relatos do uso desses processos no contexto de DDS, é importante analisá-los pois estes integram as atividades de desenvolvimento e teste de software.

O XP e o AUP utilizam a abordagem *Test Driven Development* (TDD), que é uma prática de desenvolvimento incremental, que se baseia na construção de testes de regressão automatizados para nortear as atividades de desenvolvimento (Urdangarim, 2008). Essas metodologias definem as atividades, atores, artefatos e apresentam o enfoque ágil. Entretanto, essas abordagens, também, não definem procedimentos para a especificação formal dos casos de teste.

A partir da análise da Tabela 3.2 é possível observar a inexistência de um processo que aborde a especificação formal dos casos de teste, o que é de extrema relevância no contexto de DDS por amenizar os problemas de compreensão e ambiguidade ocasionados pelas diferenças culturais e linguistícas. Dos processos analisados, nenhum contempla explicitamente o gerenciamento descentralizado, oferece mecanismos para acompanhar a evolução dos trabalhos dos times, promover a colaboração e a confiança.

No contexto de DDS, o uso de uma notação padrão facilita a comunicação entre as equipes pois, possibilita que todos os artefatos utilizados para testar um sistema possam ser especificados e modelados, auxiliando na documentação, entendimento e rastreabilidade dos artefatos.

3.5 Abordagem de Desenvolvimento e Teste de Software

Em Alves et al. (2008) é apresentada uma proposta de integração dos processos de Desenvolvimento Dirigido por Modelos (MDD) e Teste Dirigido por Modelos (MDT). O MDD objetiva mudar o foco do desenvolvimento de software para os modelos, contribuindo para aumentar a qualidade do software, facilitar a portabilidade e diminuir os custos inerentes à evolução. O MDT refere-se à geração automática de casos de teste a partir de modelos abstratos, contribuindo para aumentar a efetividade, confiabilidade e produtividade em processos de teste.

Segundo Alves et al. (2008), a integração destes processos pode trazer diversos benefícios, tais como: redução do custo de desenvolvimento; alcance de um alto nível de automação no desenvolvimento e geração de casos de testes; facilidade para realizar alterações nos requisitos, devido à rastreabilidade provida; e, menores custos de manutenção.

A abordagem proposta representa as aplicações utilizando *Model Driven Archicteture* (MDA), com modelos independentes dos detalhes de implementação (*Computational Independent Model* - CIM e *Platform Independent Model* - PIM) e dependentes da linguagem de programação (*Platform Specific Model* - PSM). Os modelos independentes de plataforma são formalizados com UML 2.0 e OCL (*Object Constraint Language*) e os dependentes utilizando Java. Para a modelagem do CIM e do PIM foram definidos os modelos funcionais, estruturais e comportamentais. A geração dos casos de testes é realizada a partir dos modelos do CIM-PIM, dando origem ao CITM (*Computational Independent Testing Models*), modelos que capturam os objetivos de teste e ao PITM (*Plataform Independent Testing Models*), modelos que refletem os casos de teste abstratos.

A especificação completa dos casos de teste selecionados e os objetivos de teste são gerados automaticamente. Por outro lado, a geração dos dados de teste é semiautomática e realizada a partir dos diagramas de classe e de máquina de estados comportamentais.

3.6 Considerações Finais

Este capítulo apresentou os trabalhos, que juntamente com os problemas existentes do DDS, deram subsídios para a elaboração da abordagem integrada de desenvolvimento e teste de software para equipes distribuídas.

A partir das análises descritas nas Seções 3.2 e 3.4 foram elencadas algumas características desejáveis para a definição de um processo de software que atenda às necessidades do desenvolvimento distribuído, sendo elas:

- gerenciamento híbrido (centralizado-descentralizado) em relação ao controle das atividades e definição de um líder para estimular a confiança e compromisso entre os membros [(Mullick et al., 2006), (Avritzer et al., 2007) e (Avritzer et al., 2008)];
- encorajamento da comunicação entre desenvolvimento e equipe de teste de integração oferecendo artefatos com informações relevantes para as duas equipes (Vanzin et al., 2005);

- especificação formal de testes para amenizar os problemas de ambiguidade e reduzir a necessidade de comunicação [(Mullick et al., 2006), (Avritzer et al., 2007) e (Avritzer et al., 2008)];
- uso de arquitetura baseada em componente para distribuição do desenvolvimento ou distribuição por fases, por exemplo desenvolvimento e teste, para reduzir a comunicação entre as equipes (Lings et al., 2007);
- identificação das equipes envolvidas, o que será realizado em cada local e as responsabilidades de cada membro, para auxiliar na percepção (Lings et al., 2007);
- adoção de metodologias de especificação de fácil compreensão e que possuam semântica para transmitir informações aos desenvolvedores, no caso a UML por ser conhecida tanto no meio acadêmico como industrial (Sengupta et al., 2006);
- integração contínua incremental [(Vanzin et al., 2005) e (Paasivaara e Lassenius, 2004)];
- desenvolvimento iterativo e com entregas frequentes para fornecer uma maior visibilidade do projeto aos gerentes [(Paasivaara e Lassenius, 2004) e (Taylor et al., 2006)];
- definição infraestrutura que possibilite a colaboração, controle da documentação e controle de versão dos artefatos [(Paasivaara e Lassenius, 2004) e (Taylor et al., 2006)]; desenvolvimento e aplicação de suítes de teste; e, estabelecimento de método para controle da documentação [(Paasivaara e Lassenius, 2004) e (Taylor et al., 2006)] ;
- definição de um idioma para formalização do processo e interação entre as equipes (Vanzin et al., 2005).

No próximo capítulo é apresentada a especificação e modelagem da abordagem de desenvolvimento e teste de software proposta para equipes distribuídas.

Capítulo

4

Abordagem de Desenvolvimento e Teste de Software para Equipes Distribuídas

Este capítulo apresenta os modelos utilizados para especificar a abordagem proposta, seguindo o metamodelo SPEM, que oferece uma estrutura de modelagem de processos baseada na UML.

Primeiramente, é descrita uma visão geral da abordagem, permitindo entender o contexto para o qual a mesma oferece suporte. Em seguida, os elementos da abordagem são detalhados.

4.1 Visão Geral da Abordagem

Um processo consiste em um conjunto de tarefas ordenadas que envolvem atividades, restrições e recursos para alcançar a saída desejada. Pode ser caracterizado por prescrever todas as suas principais atividades, utilizar recursos, estar sujeito a restrições, gerar produtos intermediários e finais (artefatos). Cada atividade apresenta critérios de entrada e saída e um conjunto de diretrizes que explicam os seus objetivos.

Conforme análise apresentada no Capítulo 3 foi possível evidenciar que há algumas lacunas no desenvolvimento distribuído de software no que se refere ao processo de desenvolvimento. Para tanto, devem ser definidas as atividades, os artefatos e os

papéis necessários em cada etapa do desenvolvimento, considerando as peculiaridades do desenvolvimento com equipes distribuídas.

Se a atividade de teste for conduzida ao acaso, defeitos no projeto podem passar despercebidos durante o teste. Portanto, uma estratégia sistemática deve ser estabelecida para o teste de software. Desse modo, apresenta-se uma abordagem integrada de desenvolvimento e teste de software que tem por objetivo atender às características das equipes distribuídas no que se refere à comunicação e coordenação, não se restringindo a domínios específicos de aplicação. Ou seja, as técnicas definidas na abordagem podem ser utilizadas em conjunto com técnicas que atendam às necessidades do domínio da aplicação, por exemplo sistemas de tempo real, que demanda especificações formais que podem ser utilizadas como complemento à abordagem proposta. A abordagem proposta considera o processo de testes como um conjunto de tarefas à parte que pode ser aplicado ao longo do processo de desenvolvimento, instanciado em paralelo.

Segundo a ISO/IEC 12207 a abordagem pode ser classificada como sendo um processo fundamental relacionado ao desenvolvimento. No entanto, apresenta alguns elementos de processos organizacionais e de apoio. Em relação aos processos organizacionais contempla aspectos de gerência. E, no que se refere aos processos de apoio, engloba itens de verificação e validação, os quais fornecem subsídios para a qualidade do produto a ser desenvolvido.

A abordagem de desenvolvimento utiliza a *Unified Modeling Language* (UML), que é uma linguagem de modelagem que permite aos desenvolvedores visualizar os produtos de seu trabalho em diagramas padronizados. O uso de especificações UML é atrativa por possuir informações relevantes para testes e estar em uso na academia e na maioria das empresas de tecnologia, de modo que não há custos extras associados ao treinamento e integração no processo de desenvolvimento de uma aplicação (Hartmannm et al., 2004).

Na abordagem de teste é utilizado o Perfil UML 2.0 (U2TP), que define uma linguagem para projetar, visualizar, especificar, analisar, construir e documentar os artefatos de teste de sistemas (OMG, 2005). A U2TP é uma linguagem de modelagem de testes que pode ser usada com tecnologias de componentes e linguagens orientadas a objeto, aplicadas em diversos domínios de aplicação e auxilia na documentação, entendimento e ratreabilidade dos artefatos de teste. Em sua concepção foram utilizados dois princípios de projeto: integração com UML e reuso. A integração com a UML refere-se ao fato do perfil ser baseado no meta-modelo UML, desse modo usando os mesmos princípios e elementos dessa linguagem. O reuso está relacionado à utilização, onde possível, dos conceitos e elementos já presentes em UML. Outra vantagem da U2TP é a definição de um conjunto de conceitos próprios à área de testes que podem ser representados e mapeados para

diferentes ferramentas de testes, como por exemplo: JUnit (teste unitário) e TTCN-3 (teste de integração e sistema). Os artefatos de teste especificados pela abordagem, são os recomendados pela norma IEEE 829, que descreve um conjunto básico de documentos de teste de software (IEEE, 1998).

A visão geral da abordagem integrada, que mostra o fluxo de informações entre as Disciplinas, é apresentada na Figura 4.1. Cabe ressaltar, que os artefatos gerados são passíveis de refinamento nas Disciplinas posteriores.

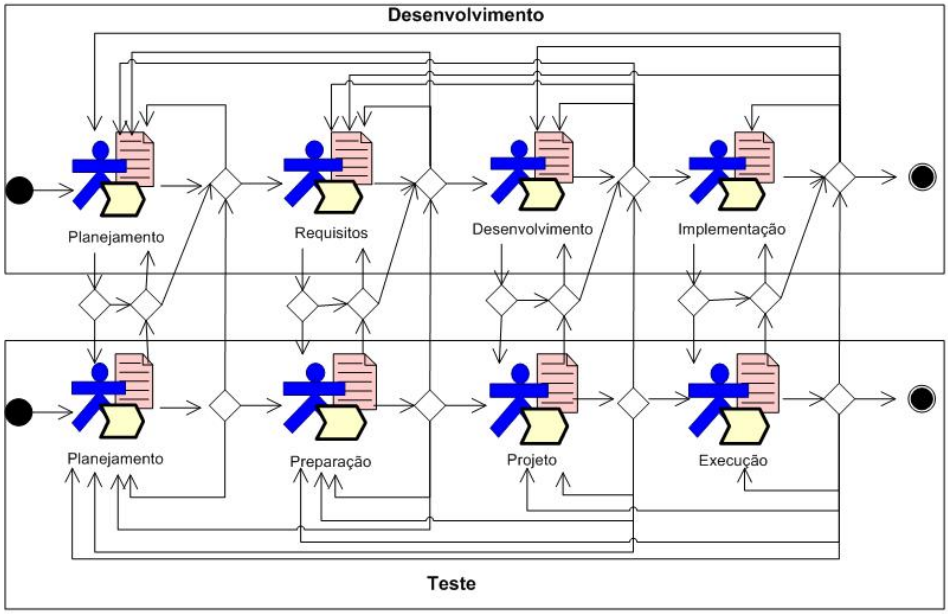

Figura 4.1: Diagrama de Atividades da Abordagem Integrada de Desenvolvimento e Teste de Software.

A integração das unidades, independente da granularidade da distribuição (disciplina, componente, caso de uso, classe ou método), deve ser sempre acompanhada, podendo ocorrer dentro de cada Disciplina, como também, entre as Disciplinas.

Segundo Alves et al. (2008), a integração dos processos de desenvolvimento e teste pode trazer diversos benefícios, dentre eles: redução do custo de desenvolvimento; alcance de um alto nível de automação no desenvolvimento e geração de casos de testes; facilidade para realizar alterações nos requisitos, devido à rastreabilidade provida; e, menores custos de manutenção.

As seções seguintes apresentam os aspectos estruturais que organizam o conteúdo das abordagens de desenvolvimento e teste em termos de atividades, artefatos e papéis envolvidos. Ressalta-se, ainda, que as abordagens de desenvolvimento e teste devem ser instanciadas em paralelo.

4.2 Abordagem de Desenvolvimento

A Figura 4.2 ilustra a visão geral da abordagem de desenvolvimento, em que é possível visualizar a ordem de execução das Disciplinas.

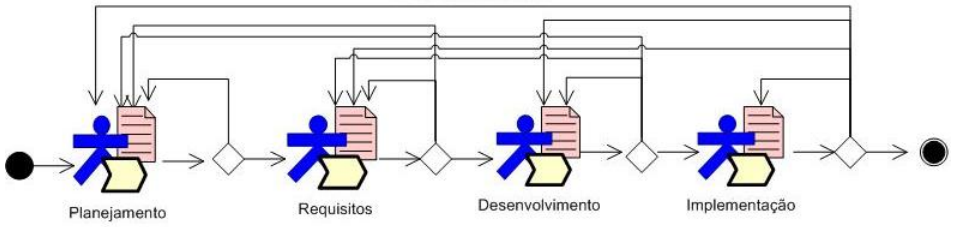

Figura 4.2: Diagrama de Atividades da Abordagem de Desenvolvimento.

Outra forma de representar a abordagem é através do conjunto de elementos que a compõem, como pode ser visto na Figura 4.3.

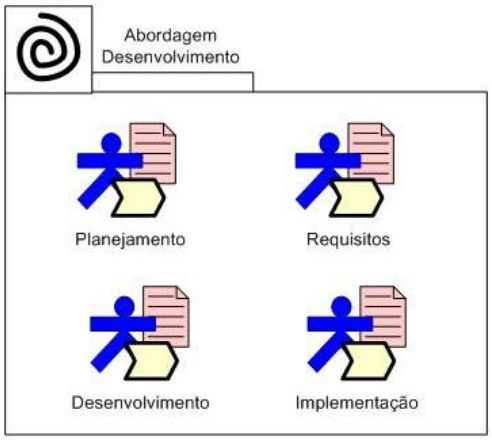

Figura 4.3: Diagrama de Pacotes da Abordagem de Desenvolvimento.

A Tabela 4.1 apresenta os papéis envolvidos, bem como sua descrição.

Tabela 4.1: Papéis na Abordagem de Desenvolvimento

Papéis da Abordagem	**Descrição**
	Responsável pelas atividades relativas ao planejamento estratégico do projeto (Enami, 2006).
	Gerencia as unidades distribuídas, distribuindo e coordenando as atividades. (Enami, 2006).
	Lidera e coordena a modelagem de casos de uso empresarial, delimita os processos de negócio que estão envolvidos. Conduz e coordena a obtenção de requisitos.
	Detalha a especificação dos requisitos do sistema.
	Lidera e coordena as atividades do projeto. Estabelece a estrutura geral de cada visão de arquitetura.
	Detalha a especificação, descrevendo o fluxo de trabalho e estrutura.
	Desenvolve a programação e realiza os testes de unidade de acordo com os padrões técnicos e tecnológicos adotados para o projeto.
	Pessoa ou entidade que solicita o desenvolvimento do software.

Os papéis identificados podem participar de dois modos distintos:

- << **perform** >>: representa que um papel executa uma atividade efetivamente.
- << **assist** >>: indica que um papel auxilia outro no desempenho de suas atividades.

As Seções seguintes descrevem cada uma das Disciplinas, sendo elas: Planejamento, Requisitos, Desenvolvimento e Implementação. Estas Disciplinas são representadas sob quatro perspectivas, as quais são fornecidas pelos Diagramas de Pacotes, Diagrama de Casos de Usos, Diagrama de Classes e Diagrama de Atividades.

4.2.1 Disciplina Planejamento

Objetivo

Essa disciplina é responsável pela configuração do processo de desenvolvimento e definição dos aspectos relacionados ao gerenciamento do projeto. A Figura 4.4 apresenta a Definição de Trabalho, mecanismo de divisão semântica para as atividades, que compõe a Disciplina.

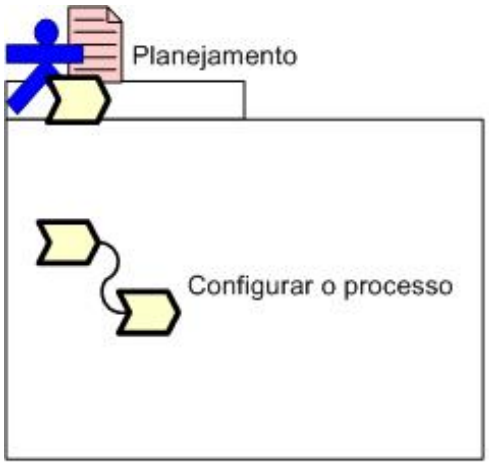

Figura 4.4: Diagrama de Pacotes da Disciplina Planejamento.

A Definição de Trabalho - Configurar o Processo - é detalhada no Diagrama de Pacotes representado na Figura 4.5, em que é possível visualizar os elementos (papéis, atividades e artefatos) que a constituem.

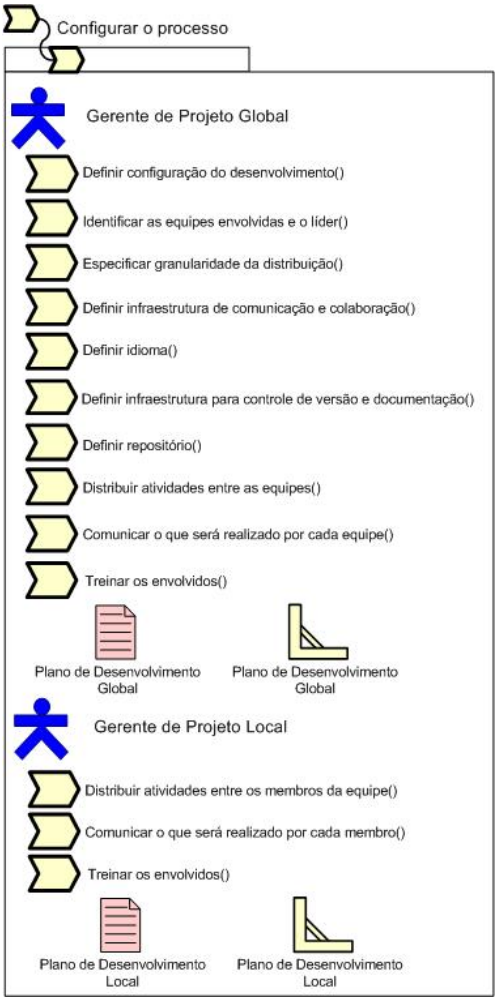

Figura 4.5: Diagrama de Pacotes da Definição de Trabalho - Configurar o processo.

Atividades

A Disciplina Planejamento é constituída pelas seguintes atividades:

- definir a configuração do desenvolvimento (*onshore insourcing, outsourcing, offshoring* ou *internal offshoring*) a ser adotada para as disciplinas posteriores. De acordo com L'Erário (2009a) é importante explorar, estrategicamente, as alianças entre organizações para obter vantagem competitiva a partir da produtividade distribuída;

- identificar as equipes envolvidas no projeto, bem como seu líder (Gerente de Projetos Local) para estimular a confiança e compromisso entre os membros. Na formação das equipes deve-se considerar, além do idioma, questões de afinidade e habilidades dos membros. Segundo Cibotto et al. (2009), sempre que possível, envolva no

projeto equipes que possuem o mesmo idioma nativo para amenizar os problemas de comunicação. Deve-se, também, estabelecer a equipe central, que será responsável pela sincronização das atividades entre as diversas equipes;

- especificar a granularidade em que será realizada a distribuição (disciplina, componente, caso de uso, classe ou método). Para estabeler a estratégia de distribuição os Gerentes de Projeto Global e Local podem considerar alguns pontos, tais como:

  1. uma arquitetura de software modular (baixo acomplamento e alta coesão) pode ser utilizada para distribuição dos esforços entre as equipes pois, reduz a complexidade e permite um desenvolvimento em paralelo simplificado (Audy e Prikladnicki, 2008);
  2. a proximidade de uma equipe do cliente pode ser considerada para a distribuição da disciplina, no caso Requisitos.
  3. as habilidades e competências das equipes envolvidas (modelagem, teste, implementação).

  Cabe ressaltar, que a estratégia de distribuição adotada tem impacto direto sobre a intensidade, frequência e o tipo de coordenação necessária entre os participantes do projeto (Sangwan et al., 2006)

- definir uma infraestrutura para comunicação e colaboração, tanto interna (equipes envolvidas no projeto) quanto externa (clientes). Para que essa comunicação ocorra de forma adequada, é necessário que a infraestrutura seja definida corretamente. Geralmente, são necessários diversos meios, tais como: viagens, telefone, e-mail, videoconferência, ferramentas de gerência de projetos on-line, *instant messengers* e *wiki* (Al-Asmari e Yu, 2006);

- definir um idioma para formalização do processo e interação entre as equipes;

- definir infraestrutura para o controle de versão dos artefatos e da documentação;

- definir um repositório comum a todas as equipes envolvidas no projeto. Os Gerentes de Projeto Global e Local devem definir os níveis de acesso de cada integrante com base nas atividades que estes desempenham e na responsabilidade que lhe é conferida;

- distribuir as atividades entre as equipes envolvidas e seus membros. Weissleder e Schlingloff (2008) apresentam duas estratégias que podem ser utilizadas para a

distribuição das atividades: minimizar a colaboração necessária entre as equipes, visto que isso minimiza os impactos negativos dos problemas de comunicação e colaboração; e, minimizar a diferença entre as equipes, reduzindo o deslocamento de tempo (fuso-horário) que afeta a comunicação síncrona ou as diferenças culturais;

- comunicar o que será realizado por cada equipe, bem como a responsabilidade de cada membro, para que todos tenham consciência do que está acontecendo, das dependências, quem está realizando determinada atividade e em que local ela está sendo executada. Audy e Prikladnicki (2008) destacam que essa consciência é importante, principalmente, sob a perspectiva da documentação e do código fonte. Segundo Carmel e Tija (2005), para aumentar a percepção podem ser utilizadas algumas técnicas, tais como, o uso de repositórios comuns, sistemas para controle de projetos, ambientes de desenvolvimento integrados, reuniões de *status*, cronograma das equipes e descrição dos processos.

- treinar os envolvidos em relação à abordagem.

Papéis

Nessa Disciplina há o Gerente de Projetos Global, que é responsável por configurar o processo de desenvolvimento por meio da realização das atividades descritas na Seção anterior e o Gerente de Projetos Local.

O relacionamento entre os papéis e atividades é ilustrado no Diagrama de Casos de Uso da Figura 4.6, em que é possível observar que o Gerente de Projeto Global é auxiliado pelos Gerentes de Projetos Local na estruturação do processo.

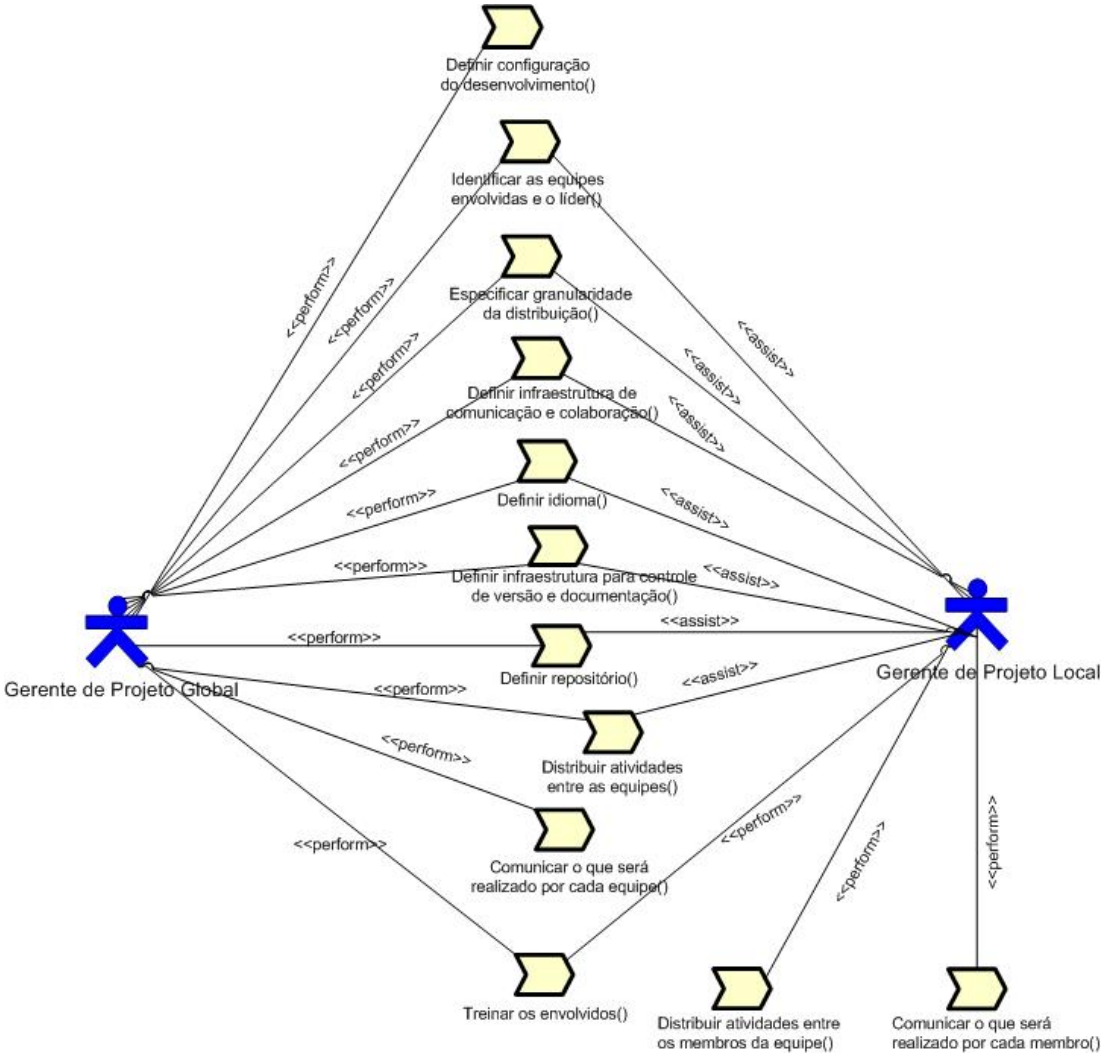

Figura 4.6: Diagrama de Casos de Uso - Definição de Trabalho - Configurar o processo.

Artefatos

Os artefatos gerados são o plano de desenvolvimento de software global e local, que fornece a linha básica de recursos e cronograma. Nesse artefato são especificadas informações sobre o escopo do projeto, restrições, estrutura organizacional, papéis e responsabilidades, recursos de hardware e software e prazos. O *template* desse artefato pode ser visto no Apêndice B.

O Diagrama de Classes da Figura 4.7 ilustra o relacionamento dos papéis e artefatos envolvidos, definindo assim, os responsáveis pela geração de cada artefato.

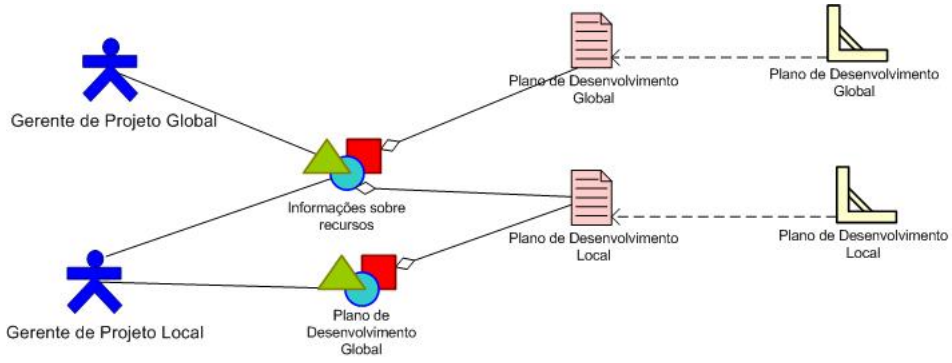

Figura 4.7: Diagrama de Classes - Definição de Trabalho - Configurar o processo.

O fluxo de atividades da Definição de Trabalho - Configurar o Processo - é representado pelo Diagrama de Atividades da Figura 4.8, que permite visualizar a precedência das atividades, bem como, as que podem ser executadas simultaneamente.

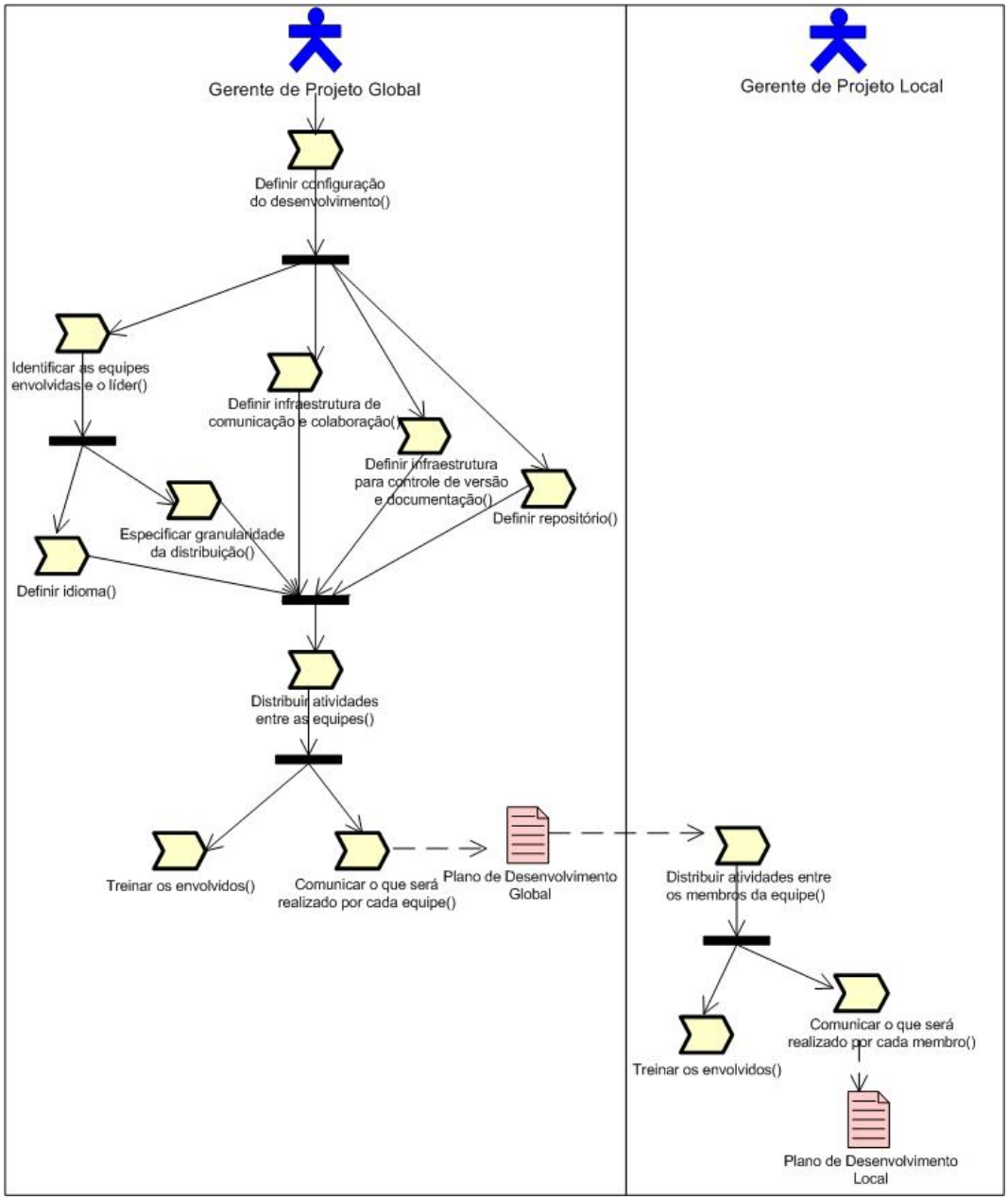

Figura 4.8: Diagrama de Atividades - Definição de Trabalho - Configurar o processo.

4.2.2 Disciplina Requisitos

Objetivo

Esta disciplina descreve como definir uma visão do sistema e traduzi-la em modelos. Como metas, tem-se: estabelecer e manter contato com os *stakeholders* sobre o que o

sistema deve fazer, facilitar o entendimento do desenvolvedor sobre os requisitos através dos modelos gerados e delimitar o escopo do sistema.

As Definições de Trabalho que compõem a Disciplina Requisitos são ilustradas na Figura 4.9.

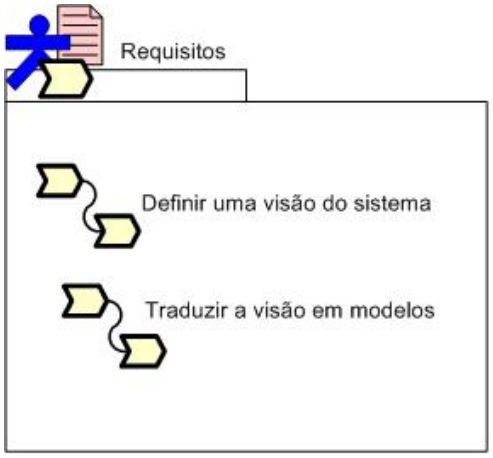

Figura 4.9: Diagrama de Pacotes da Disciplina Requisitos.

A Definição de Trabalho - Definir uma visão do sistema - é detalhada no Diagrama de Pacotes representado na Figura 4.10, por meio da apresentação dos papéis, artefatos e atividades envolvidos.

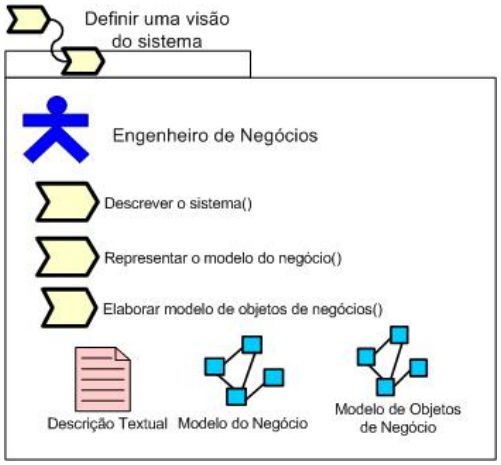

Figura 4.10: Diagrama de Pacotes da Definição de Trabalho - Definir uma visão do sistema.

Os elementos (papéis, artefatos e atividades) que compõem a Definição de Trabalho - Traduzir a visão em modelos - são apresentados no Diagrama de Pacotes da Figura 4.11.

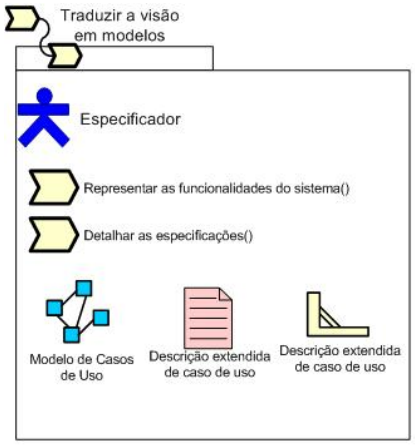

Figura 4.11: Diagrama de Pacotes da Definição de Trabalho - Traduzir a visão em modelos.

Atividades

A Disciplina Requisitos compreende a realização das seguintes atividades:

- descrever textualmente o sistema a ser desenvolvido;
- representar o modelo do negócio para entender a estrutura e a dinâmica, os problemas atuais e identificar potenciais melhorias, que irá gerar a arquitetura inicial do software (Kruchten, 2003);
- elaborar o modelo de objetos de negócio através da identificação dos itens considerados importantes, os quais formam o vocabulário do sistema a ser modelado;
- modelar casos de uso. Um caso de uso representa uma funcionalidade do sistema mostrando as interações entre ele e os atores externos. Além disso, casos de uso são fáceis de entender e têm uma estrutura simples. Por isso, eles são uma boa fonte para os requisitos de testes de sistemas (Guimarães, 2005).
- detalhar as especificações dos casos de uso por meio de restrições utilizando a *Object Constraint Language* (OCL) que é uma linguagem de expressões textuais precisas, utilizada na descrição de restrições em modelos orientados a objeto, como forma de complementar a parte gráfica dos modelos, para descrever restrições, que não conseguem ser diagramaticamente representadas. Segundo (Rorato, 2007), a OCL é uma linguagem formal baseada em modelos, que adota uma sintaxe simples, e que utiliza símbolos matemáticos mais simples da teoria de conjuntos e lógica, numa proposta de ser precisa, porém de fácil compreensão (escrita-leitura) por quem não

possui conhecimento matemático aprofundado. Um dos mais importantes aspectos de OCL é que se tornou parte do padrão *Unified Modeling Language* (UML). No contexto de DDS, o uso de OCL permite reduzir a ambiguidade gerada nos artefatos devido aos fatores culturais.

Papéis

Os papéis envolvidos nessa disciplina são o Engenheiro de Negócios e o Especificador.

As Figuras 4.12 e 4.13 apresentam os Diagramas de Casos de Uso que ilustram os relacionamentos entre os papéis e as atividades nas Definições de Trabalho, Definir uma visão do sistema e Traduzir a visão em modelos, respectivamente.

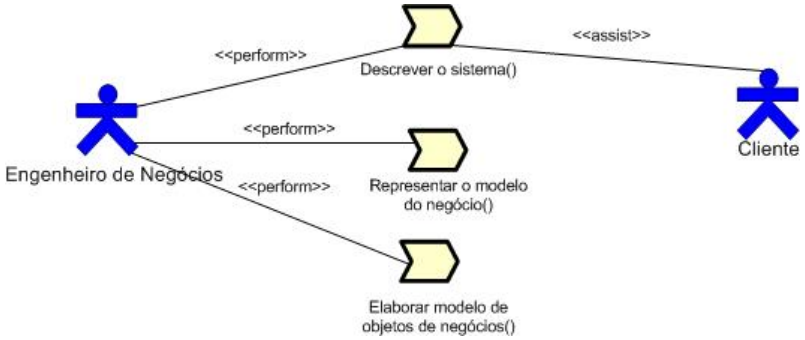

Figura 4.12: Diagrama de Casos de Uso - Definição de Trabalho - Definir uma visão do sistema.

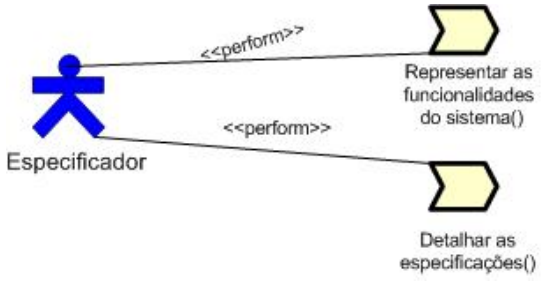

Figura 4.13: Diagrama de Casos de Uso - Definição de Trabalho - Traduzir a visão em modelos.

Artefatos

Nesta disciplina é consumido o artefato plano de desenvolvimento e são produzidas a descrição textual do sistema, modelo do negócio (arquitetura inicial), o modelo de objetos de negócio, os diagramas de casos de uso e a descrição extendida de caso de uso (vide *template* no Apêndice B).

As Figuras 4.14 e 4.15 mostram os relacionamentos entre os papéis e artefatos envolvidos em cada uma das Definições de Trabalho.

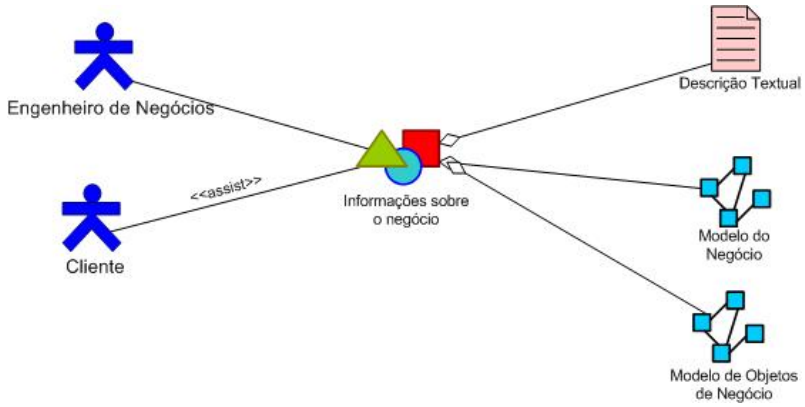

Figura 4.14: Diagrama de Classes - Definição de Trabalho - Definir uma visão do sistema.

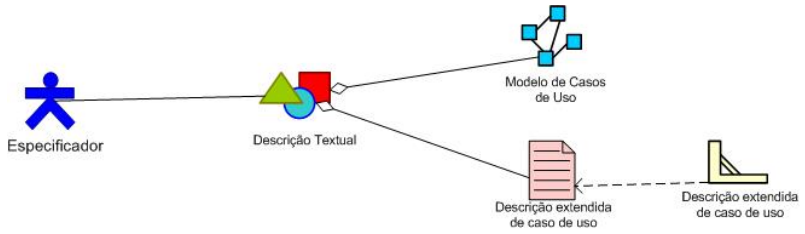

Figura 4.15: Diagrama de Classes - Definição de Trabalho - Traduzir visão em modelos.

Nas Figuras 4.16 e 4.17 são apresentados os fluxos de atividades nas Definições de Trabalho que compõem a Disciplina Requisitos.

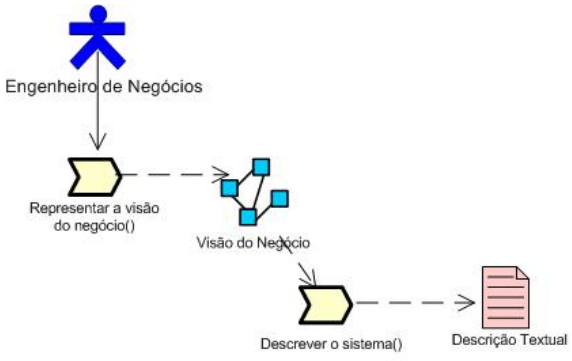

Figura 4.16: Diagrama de Atividades - Definição de Trabalho - Definir uma visão do sistema.

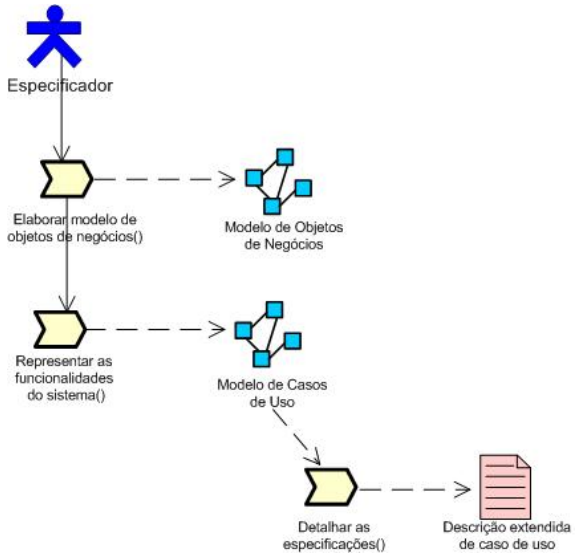

Figura 4.17: Diagrama de Atividades - Definição de Trabalho - Traduzir visão em modelos.

4.2.3 Disciplina Desenvolvimento

Objetivo

Essa disciplina objetiva traduzir os artefatos gerados na fase de Requisitos em uma especificação que descreve como implementar o sistema.

A Definição de Trabalho que compõe a Disciplina Desenvolvimento é apresentada na Figura 4.18.

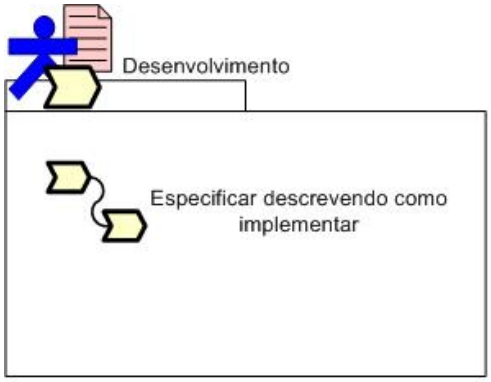

Figura 4.18: Diagrama de Pacotes da Disciplina Desenvolvimento.

A Figura 4.19 apresenta o Diagrama de Pacotes da Definição de Trabalho - Especificar descrevendo como implementar.

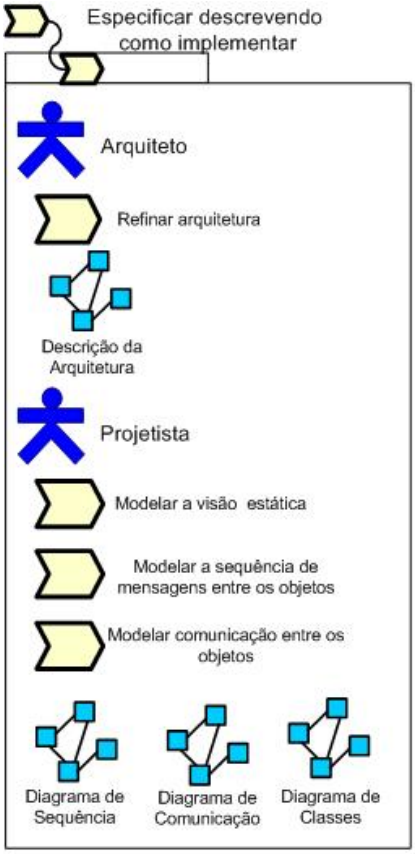

Figura 4.19: Diagrama de Pacotes da Definição de Trabalho - Especificar descrevendo como implementar.

Atividades

A Disciplina Desenvolvimento é executada por meio da realização das seguintes atividades:

- modelar a visão estática do sistema por meio da descrição de um conjunto de objetos que compartilham os mesmos atributos, operações, relacionamentos e semântica;
- modelar a sequência de mensagens entre os objetos;
- modelar comunicação entre os objetos;
- refinar descrição da arquitetura apresentada pelo modelo do negócio;

Papéis

Os papéis envolvidos nessa disciplina são o Arquiteto e o Projetista. O Diagrama de Casos de Uso que ilustra os relacionamentos entre os papéis e as atividades da Definição de Trabalho - Especificar descrevendo como implementar - é apresentado na Figura 4.20.

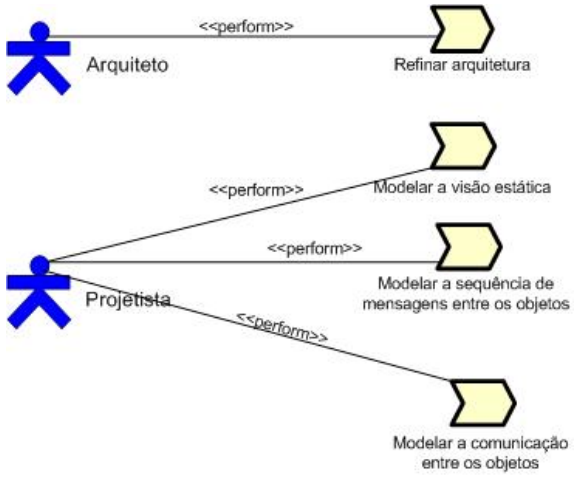

Figura 4.20: Diagrama de Casos de Uso - Definição de Trabalho - Especificar descrevendo como implementar.

Artefatos

Essa disciplina consome os artefatos gerados na disciplina de Requisitos. E produz: Diagrama de Sequência, Diagrama de Comunicação, Diagrama de Classes e Descrição da Arquitetura.

A Figura 4.21 mostra o relacionamento entre os papéis e artefatos envolvidos.

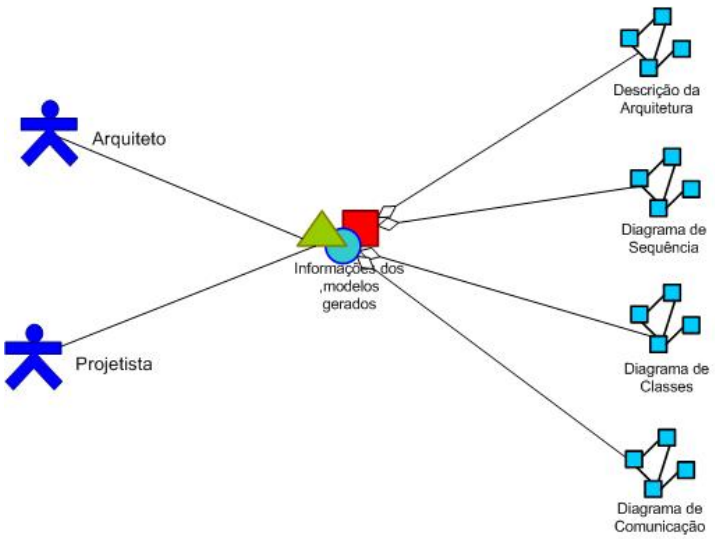

Figura 4.21: Diagrama de Classes - Definição de Trabalho - Especificar descrevendo como implementar.

O fluxo de atividades da Definição de Trabalho - Especificar descrevendo como implementar - é apresentado na Figura 4.26.

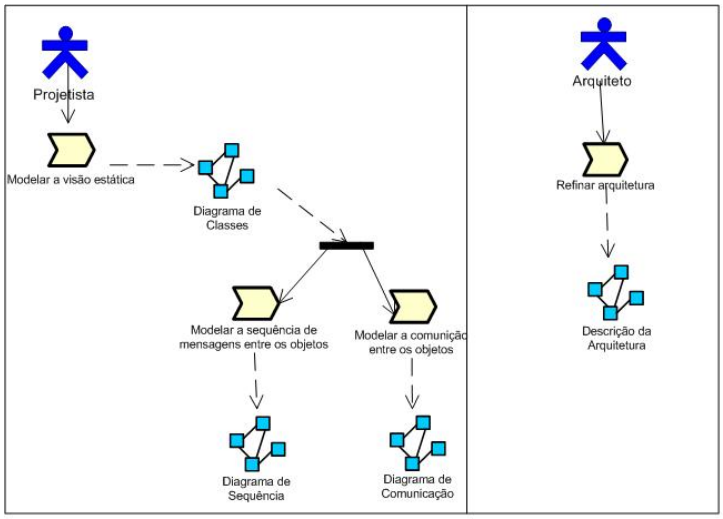

Figura 4.22: Diagrama de Atividades - Definição de Trabalho - Especificar descrevendo como implementar.

4.2.4 Disciplina Implementação

Objetivo

A Disciplina Implementação tem como metas a definição da organização do código fonte, a implementação das classes e objetos e a integração dos resultados alcançados pelas diversas equipes em sistema executável, se essa disciplina, também, for distribuída.

A Figura 4.23 apresenta a Definição de Trabalho que compõe a Disciplina Implementação.

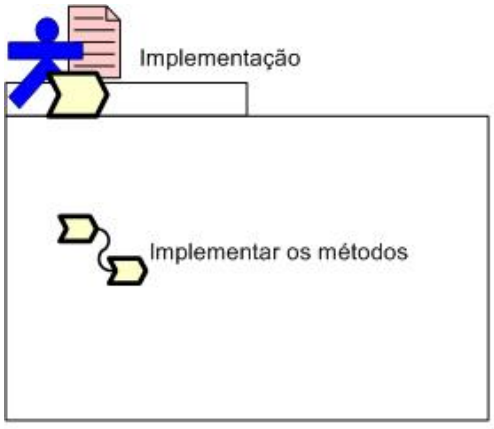

Figura 4.23: Diagrama de Pacotes da Disciplina Implementação.

O Diagrama de Pacotes da Figura 4.24 mostra os papéis, atividades e artefatos envolvidos na Definição de Trabalho - Implementar os métodos.

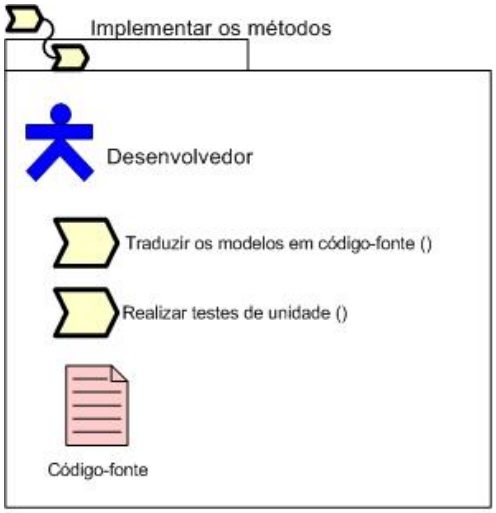

Figura 4.24: Diagrama de Pacotes da Definição de Trabalho - Implementar os métodos.

Atividades

As atividades que compõem essa disciplina são:

- traduzir os modelos gerados nas disciplinas anteriores em código fonte;
- realizar os testes de unidade;

Papéis

O Desenvolvedor é o papel que desempenha as atividades dessa Disciplina.

O Diagrama de Casos de Uso, apresentado na Figura 4.25, mostra os relacionamentos entre os papéis e as atividades.

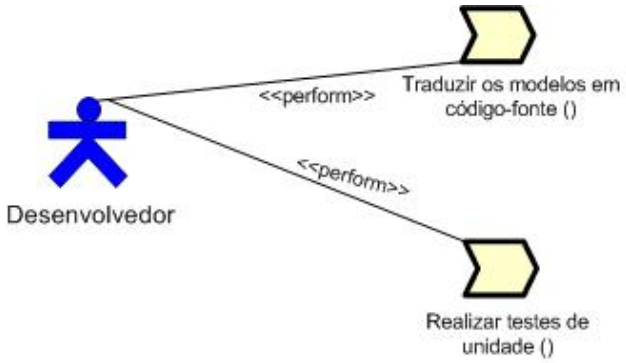

Figura 4.25: Diagrama de Casos de Uso - Definição de Trabalho - Implementar as classes e objetos.

Artefatos

Utiliza como entrada os artefatos da Disciplina de Desenvolvimento e produz o código-fonte.

A Figura 4.26 ilustra o fluxo de atividades da Definição de Trabalho - Implementar as classes e objetos.

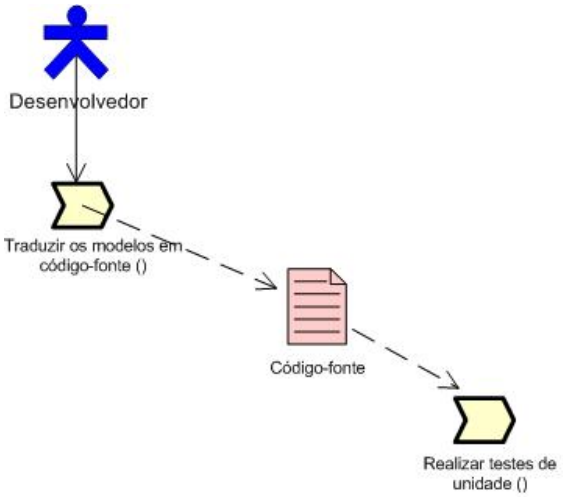

Figura 4.26: Diagrama de Atividades - Definição de Trabalho - Implementar as classes e objetos.

4.3 Abordagem de Teste

O diagrama de atividades da Figura 4.27 mostra a ordem em que disciplinas são executadas.

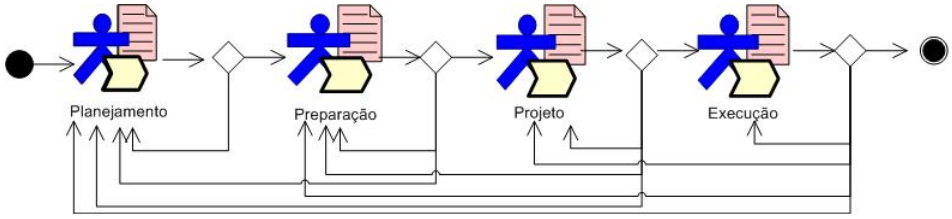

Figura 4.27: Diagrama de Atividades da Abordagem de Teste.

Na Figura 4.28 são mostrados os elementos que compõem a abordagem de teste.

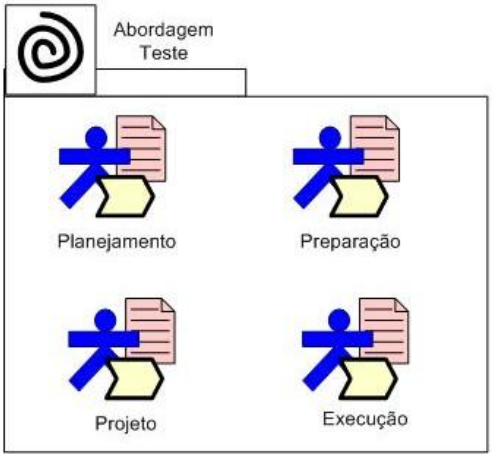

Figura 4.28: Diagrama de Pacotes da Abordagem de Teste.

Na abordagem, os papéis foram definidos com base na diretrizes do RUP. A Tabela 4.2 abaixo apresenta os papéis identicados e sua descrição.

Tabela 4.2: Papéis na Abordagem de Teste

Papéis da Abordagem	**Descrição**
	Gerencia as unidades distribuídas, distribuindo e coordenando s atividades. (Enami, 2006).
	Responsável pelo planejamento e projeto de testes.
	Responsável pela execução dos testes, avaliação da execução e dos resultados do testes.

As Seções seguintes descrevem cada uma das Disciplinas que compõem a abordagem de testes, sendo elas: Planejamento, Preparação, Projeto e Execução. Estas Disciplinas são representadas por meio de quatro diagramas: Diagramas de Pacotes, Diagrama de Casos de Usos, Diagrama de Classes e Diagrama de Atividades.

4.3.1 Disciplina Planejamento

Objetivo

É a disciplina responsável pela configuração do processo de teste, trata da definição dos fatores relacionados à gerência. A Figura 4.29 apresenta a Definição de Trabalho que compõe a Disciplina.

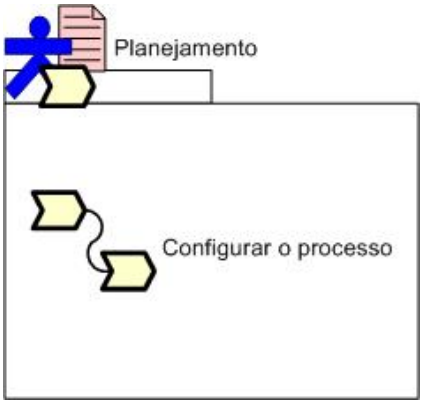

Figura 4.29: Diagrama de Pacotes da Disciplina Planejamento.

A Definição de Trabalho - Configurar o Processo - é detalhada no Diagrama de Pacotes da Figura 4.30.

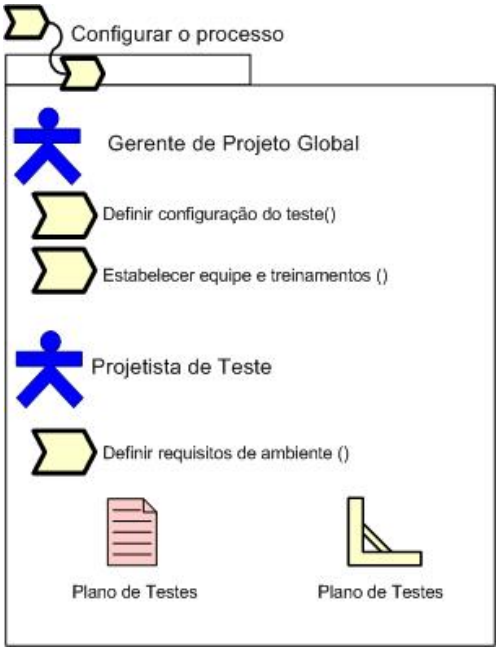

Figura 4.30: Diagrama de Pacotes da Definição de Trabalho - Configurar o processo.

Atividades

As atividades que compõem essa disciplina são:

- definir a configuração do teste (*onshore insourcing, offshoring, outsourcing* ou *internal offshorring*). Essa configuração pode ser diferente da definida na abordagem de desenvolvimento. Pode-se optar por terceirizar os testes de integração, sistemas e de aceitação para empresas que sejam especialistas na área. Segundo (Bazman, 2007), a terceirização da atividade de testes pode ser motivada pela redução de custos, velocidade, melhoria dos testes e aquisição de instalações de ambientes de teste. E como, benefícios tem-se: retorno do investimento; melhoria da confiabilidade do software; realização de uma maior gama de testes; contratação de equipe de teste eficiente e qualificada num curto período de tempo; e, maior eficiência. Em casos de terceirização os artefatos definidos na abordagem deverão ser utilizados de modo a garantir que as informações necessárias resultantes do teste serão entregues. Com isto, minimiza-se a possibilidade de eventuais impactos negativos causados pela terceirização e diferenças de cultural organizacional.
- estabelecer equipe de teste e treinamentos necessários;
- definir os requisitos de ambiente (ferramentas, pessoas e hardware).

Papéis

Os papéis envolvidos são o Gerente de Projetos Global e o Projetista de Testes. O relacionamento entre os papéis e atividades é ilustrado no Diagrama de Casos de Uso da Figura 4.31.

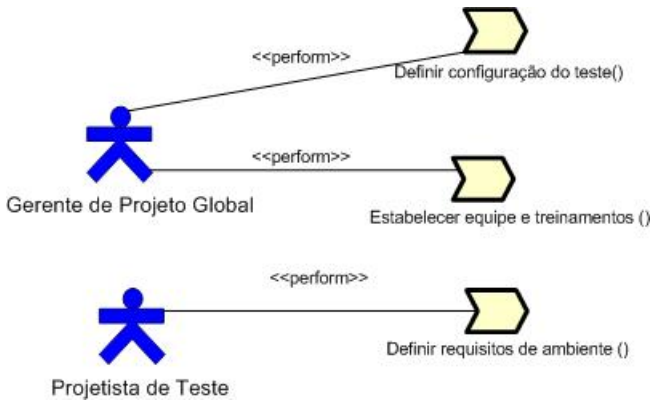

Figura 4.31: Diagrama de Casos de Uso - Definição de Trabalho - Configurar o processo.

Artefatos

O artefato consumido nessa discplina é o Plano de Desenvolvimento de Software. E, o artefato produzido é uma versão inicial do Plano de Testes.

A Figura 4.32 apresenta o Diagrama de Classes, em que é possível visualizar o relacionamento dos papéis e artefatos envolvidos, definindo assim, os responsáveis pela geração de cada artefato.

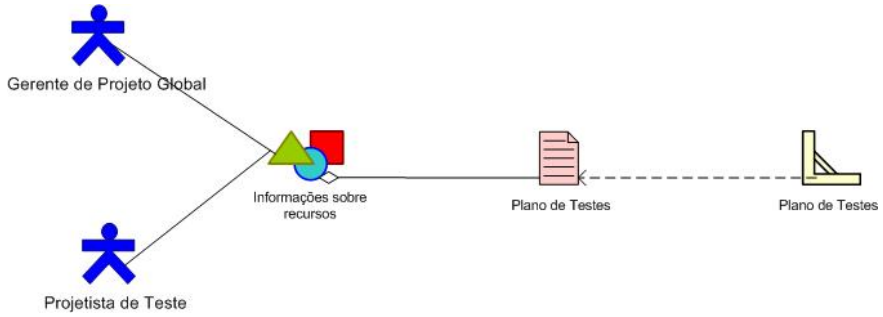

Figura 4.32: Diagrama de Classes - Definição de Trabalho - Configurar o processo.

O fluxo de atividades da Definição de Trabalho - Configurar o processo - é representado pelo Diagrama de Atividades da Figura 4.33, que permite visualizar a precedência das atividades.

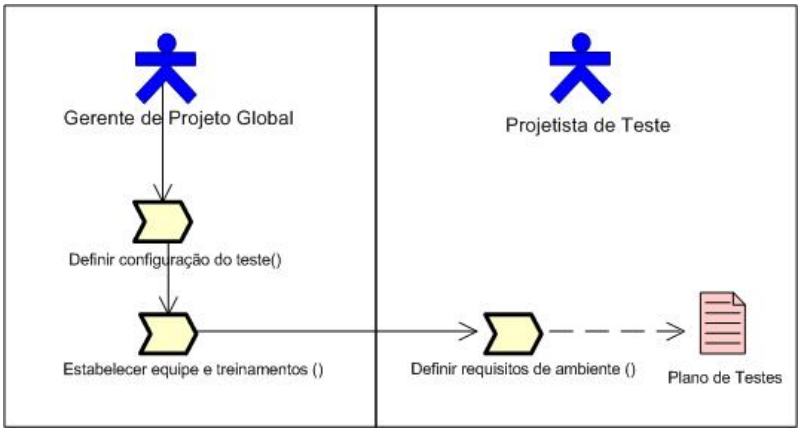

Figura 4.33: Diagrama de Atividades - Definição de Trabalho - Configurar o processo.

4.3.2 Disciplina Preparação

Objetivo

O objetivo dessa disciplina é descrever o planejamento de todas as atividades envolvidas no teste de um software. Na Figura 4.34 é apresentada a Definição de Trabalho que compõe a Disciplina.

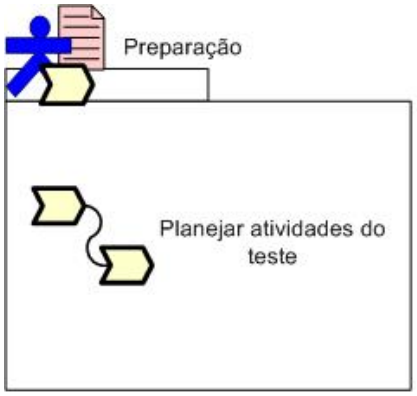

Figura 4.34: Diagrama de Pacotes da Disciplina Preparação.

A Definição de Trabalho - Planejar atividades de teste - é detalhada no Diagrama de Pacotes da Figura 4.35.

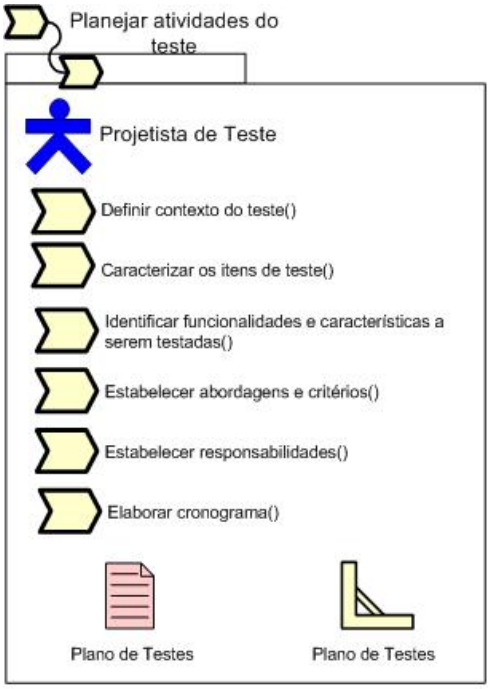

Figura 4.35: Diagrama de Pacotes da Definição de Trabalho - Planejar atividades de teste.

Atividades

A Disciplina Preparação é composta pelas seguintes atividades:

- definir o contexto do teste. Consiste em descrever, resumidamente, as funcionalidades e características a serem testadas, identificando os objetivos e o escopo do teste;
- caracterizar os itens de teste, levantando informações sobre os mesmos e descrevendo-os de forma resumida;
- identificar funcionalidades e características (por exemplo, número de acessos concorrentes e volume de dados em situações de pico) a serem testadas. As técnicas de teste baseado em risco constituem um bom indicativo na identificação de fatores de riscos associados aos requisitos do software. Essa técnica pode ser utilizada para priorizar, com base na probabilidade de ocorrência e impacto, a maior cobertura de teste em determinadas funcionalidades do sistema, uma vez que a atividade de testes é custosa, o domínio de entradas e saídas é diverso e há muitas possibilidades de caminhos a serem testados. Desse modo, pode-se priorizar esforços e alocar

recursos para as funcionalidades que necessitam ser testadas cuidadosamente (Souza e Gusmão, 2008);

- estabelecer abordagens e critérios. Esta atividade engloba a determinação da abordagem geral do teste, definição das atividades, técnicas e critérios para testar, seleção de critérios adicionais de término com base na cobertura e abrangência, determinação dos critérios de aprovação/reprovação de cada item de teste e determinação dos critérios de suspensão e os requisitos para retomada de teste (Souza e Gusmão, 2008). A partir dos diagramas de casos de uso, podem ser aplicados seis critérios de teste (Carniello, 2003):

  - **c**ritérios todas-as-comunicações, todas-as-inclusões e todas-as-extensões, que requerem o exercício de todos os relacionamentos de cada tipo;
  - **c**ritérios todas-as-comunicações-inclusões-extensões que requer o exercício de todos os relacionamentos de um diagrama;
  - **c**ritérios todos-os-estendidos-combinações e todos-os-extensores-combinaçoes, que requerem o exercício e o não-exercício dos relacionamentos de extensão.

  A especificação detalhada dos casos de uso possibilita derivar casos de teste da descrição de fluxo esperado, do fluxo alternativo e dos itens requeridos (Jacobson, 1992);

- estabelecer responsabilidades, determinando os responsáveis pelas atividades de teste;

- elaborar cronograma. Consiste em definir os marcos de teste, estimar o tempo e determinar os prazos para executar cada atividade e utilizar os recursos.

Papéis

O papel envolvido na execução dessa disciplina é o de Projetista de Teste. O Diagrama de Casos de Uso da Figura 4.36 mostra o relacionamento entre o papel e as atividades.

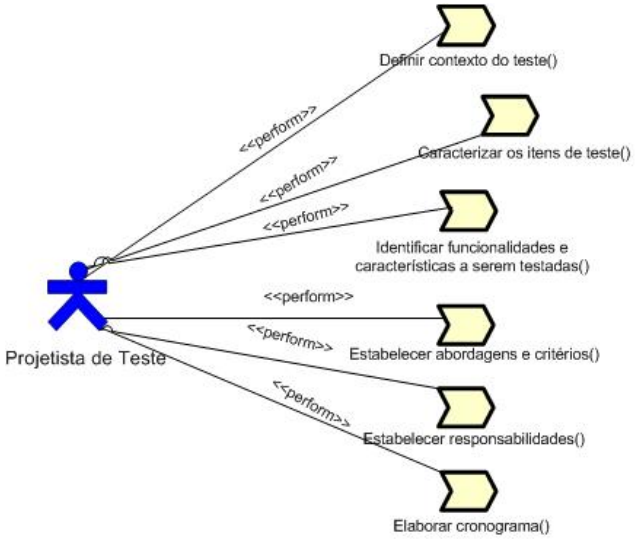

Figura 4.36: Diagrama de Casos de Uso - Definição de Trabalho - Planejar atividades de teste.

Artefatos

Essa disciplina utiliza como entrada os artefatos gerados na disciplina de Requisitos. E gera como artefato uma atualização do Plano de Teste contendo o planejamento para execução do teste, incluindo a abrangência, abordagem, recursos e cronograma das atividades de teste. Identifica os itens e as funcionalidades a serem testadas e as tarefas a serem realizadas.

A Figura 4.37 apresenta o Diagrama de Classes, em que é possível visualizar o relacionamento dos papéis e artefatos envolvidos.

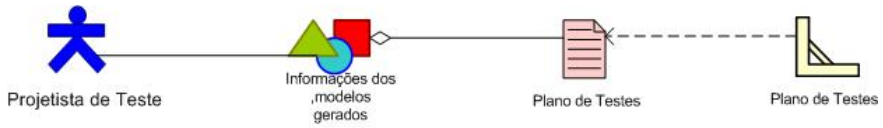

Figura 4.37: Diagrama de Classes - Definição de Trabalho - Planejar atividades de teste.

O fluxo de atividades da Definição de Trabalho - Planejas atividades de teste - é representado pelo Diagrama de Atividades da Figura 4.38, que permite visualizar a precedência das atividades.

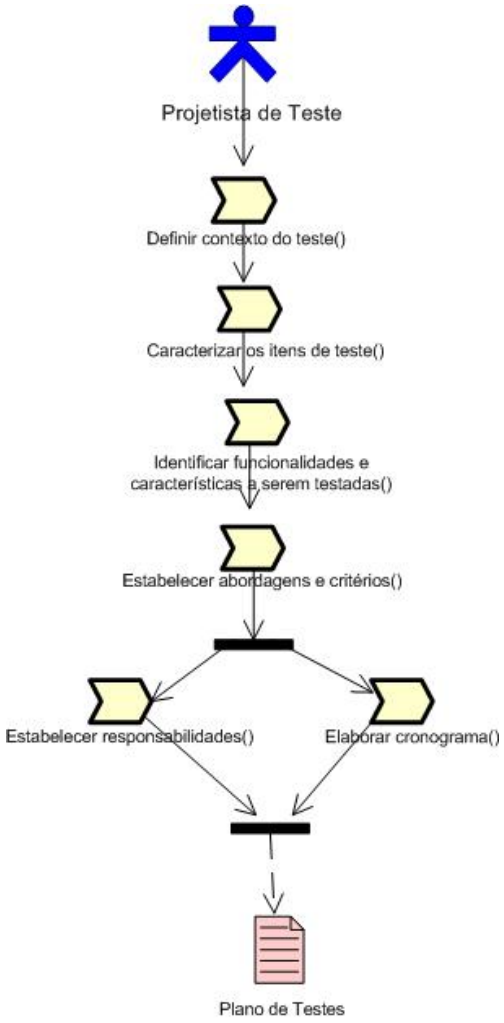

Figura 4.38: Diagrama de Atividades - Definição de Trabalho - Planejar atividades de teste.

4.3.3 Disciplina Projeto

Objetivo

A Disciplina de Projeto objetiva detalhar os aspectos técnicos a serem adotados na condução das atividades de teste. A Figura 4.39 mostra o Diagrama de Pacotes em que é possível visualizar a Definição de Trabalho que compõe a Disciplina.

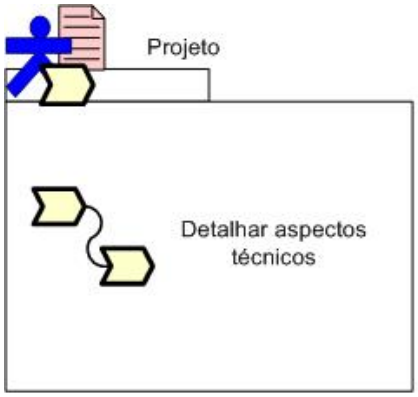

Figura 4.39: Diagrama de Pacotes da Disciplina Projeto.

Os elementos que compõem a Definição de Trabalho - Detalhar Aspectos Técnicos - são apresentados no Diagrama de Pacotes da Figura 4.40.

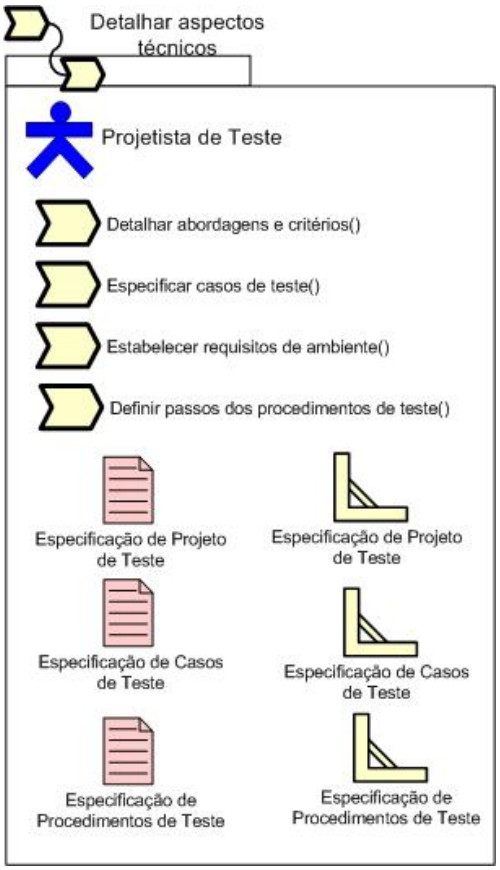

Figura 4.40: Diagrama de Pacotes da Definição de Trabalho - Detalhar aspectos técnicos.

Atividades

As atividades que constituem essa disciplina são:

- detalhar abordagens e critérios de teste;
- especificar casos de teste. As restrições OCL realizadas nos métodos das classes podem ser utilizadas como oráculo (Packevicius, 2007). O Diagrama de Sequências pode ser convertido em uma árvore de teste e cada caminho corresponde a um caso de teste;
- estabelecer requisitos de ambiente de teste com base na infraestrutura necessária;
- definir os passos dos procedimentos de teste.

Papéis

O papel envolvido é o Projetista de Teste. O Diagrama de Casos de Uso da Figura 4.41 ilustra o relacionamento entre o Projetista de Teste e as atividades dessa Disciplina.

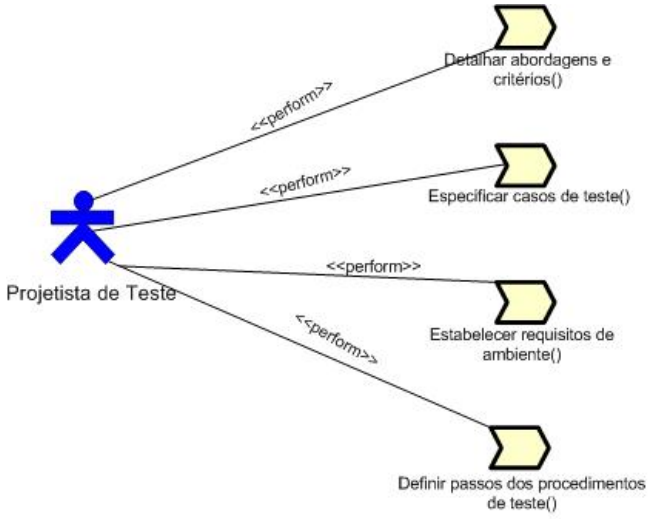

Figura 4.41: Diagrama de Casos de Uso - Definição de Trabalho - Detalhar aspectos técnicos.

Artefatos

Esta disciplina utiliza como entrada os gerados na disciplina de Desenvolvimento e os da Disciplina de Preparação. E produz como artefatos:

- Especificação de Projeto de Teste: contém um refinamento da abordagem apresentada no Plano de Teste e identifica as funcionalidades e características a serem

testadas pelo projeto e por seus testes associados, identifica os casos e os procedimentos de teste;

- Especificação de Casos de Teste: define os casos de teste, incluindo dados de entrada, resultados esperados, ações e condições gerais para a execução do teste;
- Especificação de Procedimentos de Teste: especifica os passos para executar um conjunto de casos de teste.

O Apêndice C apresenta o *template* para cada um desses artefatos.

A Figura 4.42 apresenta o Diagrama de Classes, em que é possível visualizar o relacionamento dos papéis e artefatos envolvidos.

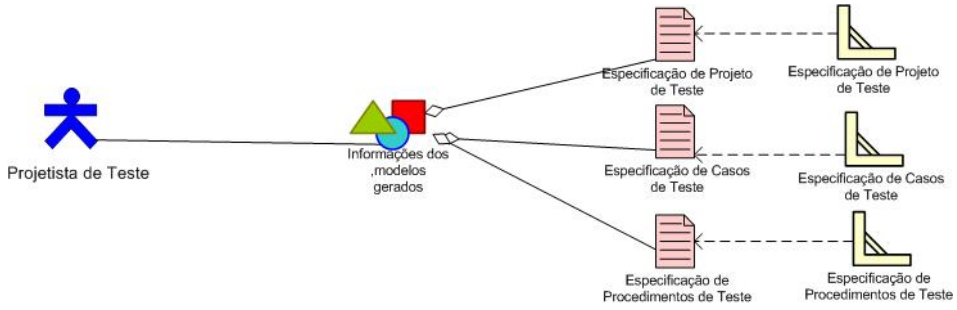

Figura 4.42: Diagrama de Classes - Definição de Trabalho - Detalhar aspectos técnicos.

O fluxo de atividades da Definição de Trabalho - Detalhar aspectos técnicos - é representado pelo Diagrama de Atividades da Figura 4.38, que permite visualizar a precedência das atividades.

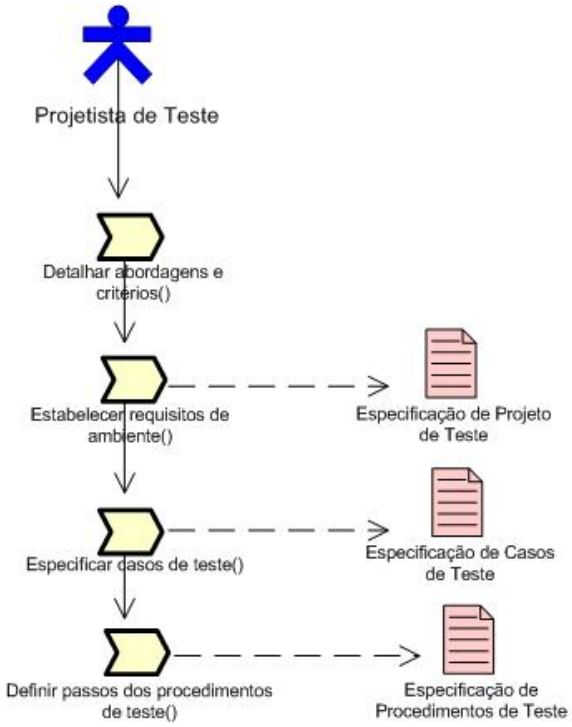

Figura 4.43: Diagrama de Atividades - Definição de Trabalho - Detalhar aspectos técnicos.

4.3.4 Disciplina Execução

Objetivo

Esta disciplina tem por objetivo a execução e registro das atividades de teste projetadas nas disciplinas anteriores. A Definição de Trabalho que compõe a Disciplina Execução é apresentada na Figura 4.44.

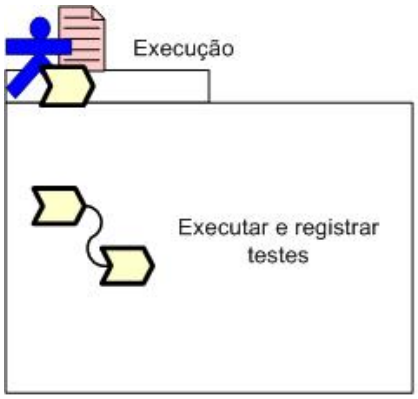

Figura 4.44: Diagrama de Pacotes da Disciplina Execução.

A Figura 4.45 apresenta o Diagrama de Pacotes da Definição de Trabalho - Executar e registrar testes.

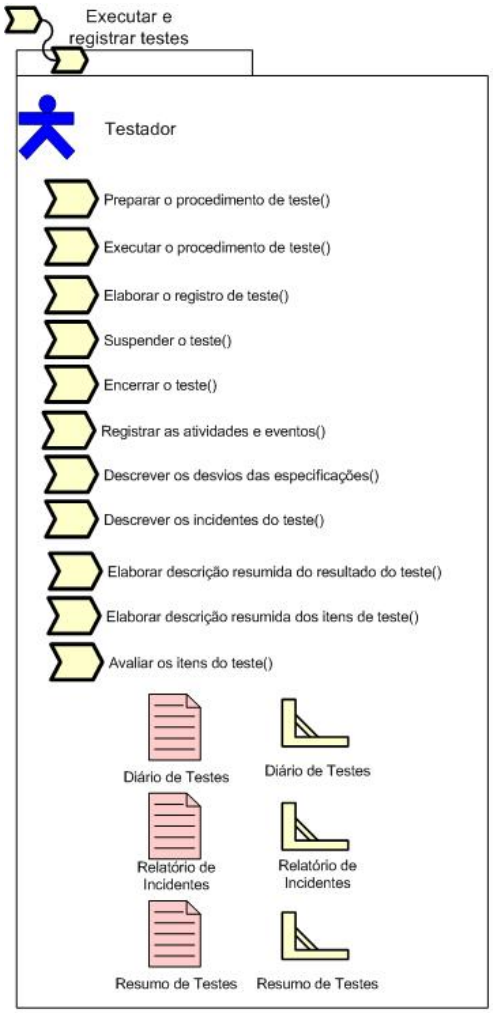

Figura 4.45: Diagrama de Pacotes da Definição de Trabalho - Executar e registrar testes.

Atividades

A Disciplina Execução constitui-se das seguintes atividades:

- preparar o procedimento de teste, que consiste na sequência de passos e ações necessárias para a execuçao de um conjunto de casos de teste relacionados (Crespo e Jino, 2005);
- executar o procedimento de teste;
- elaborar o registro de teste com resultado da execução do teste;

- suspender o teste;
- encerrar o teste;
- registrar as atividades e eventos;
- descrever os incidentes do teste, ou seja, qualquer evento que ocorra durante a execução do teste e que requeira investigação, tais como: defeito no software ou anomalia de funcionamento do ambiente;
- elaborar descrição resumida dos itens de teste;
- descrever os desvios das especificações, ou seja, as discrepâncias dos itens de teste em relação às especificações;
- elaborar descrição resumida do resultado do teste;
- avaliar os itens de teste.

Papéis

O papel envolvido nessa disciplina é o de Testador.

O Diagrama de Casos de Uso que ilustra os relacionamentos entre o papel Testador e as atividades da Definição de Trabalho - Executar e registrar teste - é apresentado na Figura 4.46.

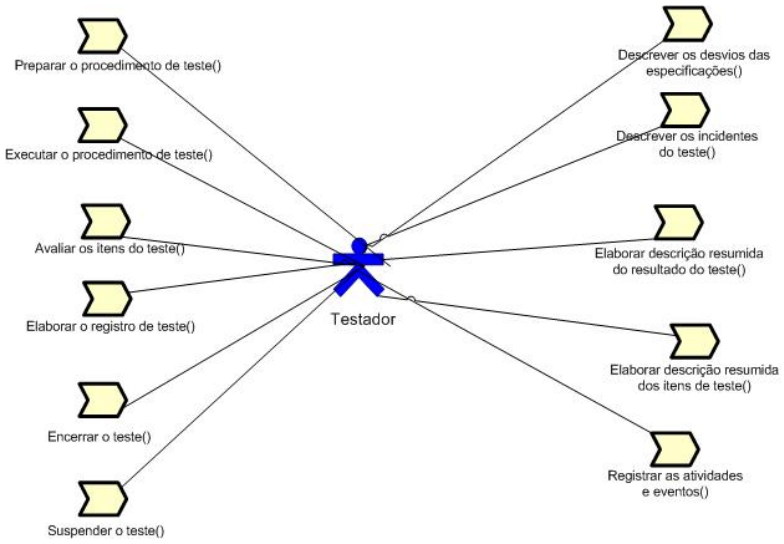

Figura 4.46: Diagrama de Casos de Uso - Definição de Trabalho - Executar e registrar teste.

Artefatos

Os artefatos consumidos por essa disciplina são os gerados nas Disciplinas de Implementação e Projeto, e são produzidos os seguintes artefatos:

- Diário de Testes: apresenta registros cronológicos dos detalhes relevantes relacionados com a execução dos testes;
- Relatório de Incidentes: documenta qualquer evento que ocorra durante a atividade de teste e que requeira análise posterior;
- Resumo de Tese: apresenta o resumo dos resultados das atividades de teste associadas com uma ou mais especificações de projeto de teste e provê avaliações baseadas nesses resultados.

No Apêndice C são apresentados os *templates* dos artefatos descritos.

A Figura 4.47 mostra o relacionamento entre os papéis e artefatos envolvidos.

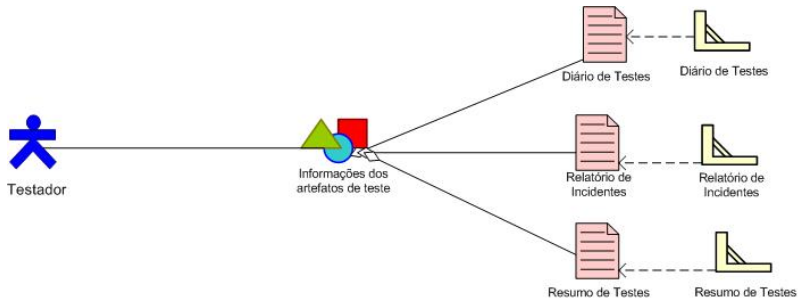

Figura 4.47: Diagrama de Classes - Definição de Trabalho - Executar e registrar teste.

O fluxo de atividades da Definição de Trabalho - Executar e registrar teste - é apresentado na Figura 4.48.

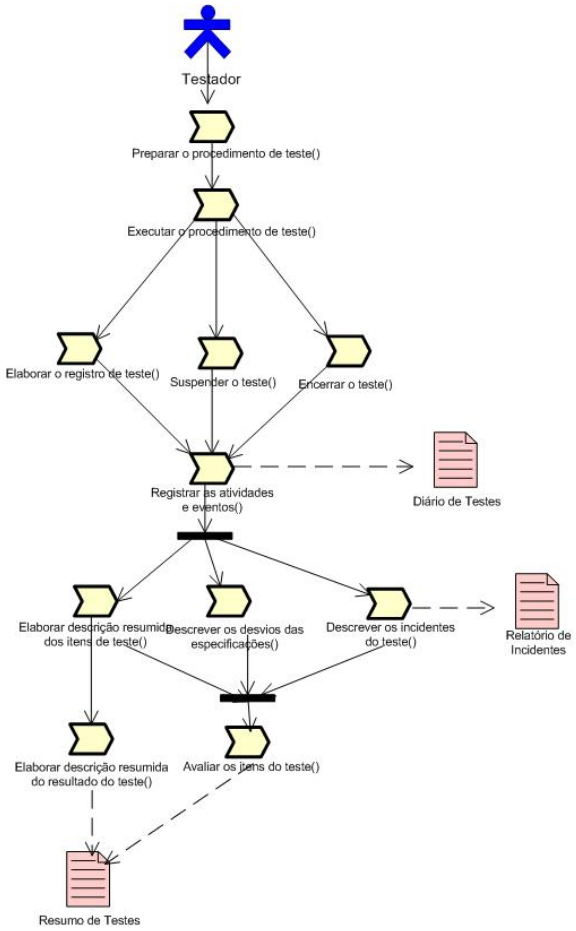

Figura 4.48: Diagrama de Atividades - Definição de Trabalho - Executar e registrar teste.

4.4 Considerações Finais

A abordagem abrange um conjunto de atividades que devem ocorrer ao longo do ciclo de vida de um projeto e as características essenciais de cada uma. Considera atividades desde o modelo de negócios a ser adotado para atuar em DDS (empresas globais, parcerias estratégicas, entre outros) até a implementação, fornecendo aos membros das equipes uma linguagem comum para definição das tarefas, atividades e marcos do projeto, o que auxilia no processo de coordenação. Além disso, oferece mecanismos para a padronização e redução da imprecisão dos artefatos, o que reduz e facilita a comunicação entre as equipes.

Capítulo

5

Exemplo de Aplicação da Abordagem

Este capítulo apresenta um exemplo de aplicação da abordagem descrita no Capítulo 4 e serve como prova de conceito. Esse tipo de estudo é realizado por meio da descrição do funcionamento da abordagem em um determinado exemplo, que pode ser real ou não. A prova de conceito caracteriza-se pela falta de formalização durante a realização do estudo, o autor não utiliza os conceitos de experimentação, o que inviabiliza obter conclusões sobre a real viabilidade e funcionalidade da abordagem (Kitchenham et al., 2002).

5.1 Cenário

Para efeitos de instanciação da abordagem foi considerado um projeto de desenvolvimento de uma aplicação web para gerenciar o processo de submissão e avaliação de artigos em eventos científicos. Devido a limitações de tempo, não foi possível instanciar a abordagem em um projeto real. Desse modo, optou-se pela realização de uma prova de conceito.

As sessões seguintes descrevem o instaciamento da abordagem para o cenário proposto. São apresentadas cada uma das disciplinas de desenvolvimento e teste.

5.2 Disciplina Planejamento - Desenvolvimento

Na execução dessa disciplina foram definidos os aspectos relacionados ao gerenciamento do projeto, em que foram alocadas duas equipes, uma localizada em Maringá-PR e a outra em Belo Horizonte-MG. Na definição da estratégia de distribuição foram considerados os

critérios de proximidade com o cliente e as habilidades e competências dos membros das equipes.

Devido à proximidade do cliente, não houve a necessidade de definição de uma infraestrutura de apoio a comunicação, sendo utilizadas reuniões presenciais. Para a comunicação entre as duas equipes foram utilizadas as seguintes ferramentas: Windows Live Messenger, Skype, e-mail e o DotProject. Como não há diferença de idioma entre as equipes envolvidas, foi adotado o Português para interação. A comunicação da abordagem a todos os membros das equipes envolvidas ocorreu por meio da documentação.

A seguir (Tabelas 5.1, 5.2 e 5.3) são apresentados o Plano Global e os Planos Locais de Desenvolvimento de Software.

Tabela 5.1: Plano Global de Desenvolvimento

Identificação do Projeto: MSPaper
Finalidade: Desenvolver uma aplicação web para apoiar as atividades relacionadas a submissão e revisão de artigos de eventos científicos.
Escopo: Este documento descreve o plano geral a ser usado pelo projeto de desenvolvimento da aplicação MSPaper. Fornecendo a todos os membros das equipes envolvidas visibilidade adequada sobre o projeto.
Restrições: (i) Número de envolvidos no projeto; (ii) Prazo;
Site Central: Maringá **Idioma:** Português **Fuso-horário:** Brasília
Gerente de Projetos Global: Gislaine Camila Lapasini Leal
Idioma do Projeto: Português
Configuração do Desenvolvimento: *onshore insourcing*
Estratégia de Distribuição: Disciplina
Sites Envolvidos: Maringá e Belo Horizonte
Estrutura Organizacional: 2 analistas e 1 desenvolvedor.
Papéis e responsabilidades: **Site Maringá:** Requisitos (Desenvolvimento) Desenvolvimento (Desenvolvimento) **Site Belo Horizonte:** Implementação (Desenvolvimento)
Recursos **Hardware:** (i) Computador pessoal dos participantes; (ii) Servidor para a ferramentade de Gerência de Projetos **Software:** (i) DotProject - ferramenta para gestão do projeto (ii) Jude - ferramenta para modelagem UML; (iii) Netbeans 6.8 - ambiente de desenvolvimento; (iv) Postgresql - sistema de gerenciamento de banco de dados
Ferramentas de comunicação: (i) DotProject; (ii) Windows Live Messenger; (iii) e-mail; (iv) Skype
Cronograma: Data de início: 22/02/2010 Data de término: 20/03/2010

Tabela 5.2: Plano Local de Desenvolvimento - Site Maringá

Escopo: Este documento descreve o plano geral a ser usado pela equipe local no desenvolvimento da aplicação MSPaper.
Idioma do Projeto: Português
Gerente de Projetos Local: Gislaine Camila Lapasini Leal
Disciplina: Requisitos (Desenvolvimento) Desenvolvimento (Desenvolvimento)
Papéis e responsabilidades: **Gislaine Camila Lapasini Leal** **Gustavo Sato**
Recursos **Hardware:** (i) Computador pessoal dos participantes; (ii) Servidor para a ferramentade Gerência de Projetos **Software:** (i) DotProject - ferramenta para gestão do projeto (ii) Jude - ferramenta para modelagem UML;
Ferramentas de comunicação: (i) DotProject; (ii) Windows Live Messenger; (iii) e-mail; (iv) Skype
Cronograma: Data de início: 25/02/2010 Data de término: 01/03/2010

Tabela 5.3: Plano Local de Desenvolvimento - Belo Horizonte

Escopo: Este documento descreve o plano geral a ser usado pela equipe local na implementação da aplicação MSPaper.
Idioma do Projeto: Português
Gerente de Projetos Local: César Alberto da Silva
Disciplina: Implementação (Desenvolvimento)
Papéis e responsabilidades: **César Alberto da Silva** Desenvolvedor: geração de código e teste de unidade.
Recursos **Hardware:** (i) Computador pessoal dos participantes; **Software:** (iii) Netbeans 6.8 - ambiente de desenvolvimento; (iv) Postgresql - sistema de gerenciamento de banco de dados (iv) JSFUnit - teste de unidade
Ferramentas de comunicação: (i) DotProject; (ii) Windows Live Messenger; (iii) e-mail; (iv) Skype
Cronograma: Data de início: 01/03/2010 Data de término: 12/03/2010

5.3 Disciplina Planejamento - Teste

Nessa disciplina foi definido o modelo de negócio da abordagem de teste, que é *onshore insourcing*. A equipe responsável por todos as disciplinas dessa abordagem está localizada em Maringá. Também, foram definidas as ferramentas e pessoas envolvidas na execução. Na Tabela 5.4 é apresentada a versão inicial do Plano de Teste.

Tabela 5.4: Plano de Teste - versão inicial

Identificação do Projeto: MSPaper
Restrições: (i) Número de envolvidos no projeto; (ii) Prazo;
Configuração do Teste: *onshore insourcing*
Estratégia de Distribuição: Disciplinas
Sites Envolvidos: Maringá
Papéis e responsabilidades: **Site Maringá:** Planejamento (Teste) Preparação (Teste) Projeto (Teste) Execução (Teste)
Recursos **Hardware:** (i) Computador pessoal dos participantes; (ii) Servidor para a ferramentade Gerência de Projetos

5.4 Disciplina Requisitos - Desenvolvimento

Na execução dessa disciplina foi estabelecido contato com o *stakeholder* (Profa. Elisa H. M. Huzita). Como a equipe responsável por essa disciplina está localizada na mesma cidade do *stakeholder*, foram realizadas duas reuniões presenciais para coleta dos requisitos. Os papéis de Engenheiro de Negócios e Especificador foram desempenhados pelo mesmo ator. Com base nos dados obtidos pelo Engenheiro de Negócios, na entrevista, foi elaborada a descrição textual do MSPaper, que pode ser vista no Apêndice D. A partir da descrição textual, foram gerados os demais artefatos, que são apresentados a seguir.

Na Figura 5.1 é ilustrada a visão de negócios da gestão de eventos.

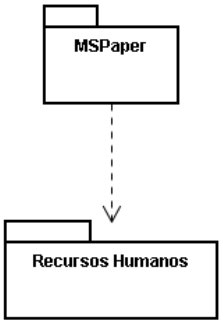

Figura 5.1: Visão do Negócio.

O modelo de objetos de negócio para o MSPaper é apresentado na Figura 5.2.

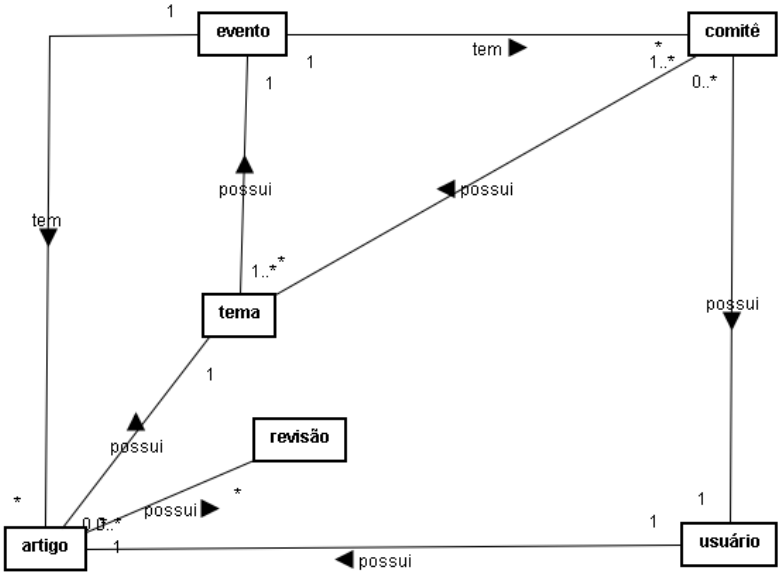

Figura 5.2: Modelo de Objetos de Negócios.

A visão estática do MSPaper é representada através dos casos de uso, atores e relacionamentos identificados, os quais são ilustrados na Figura 5.3. No Apêndice D é apresentada a especificação estendida dos casos de uso.

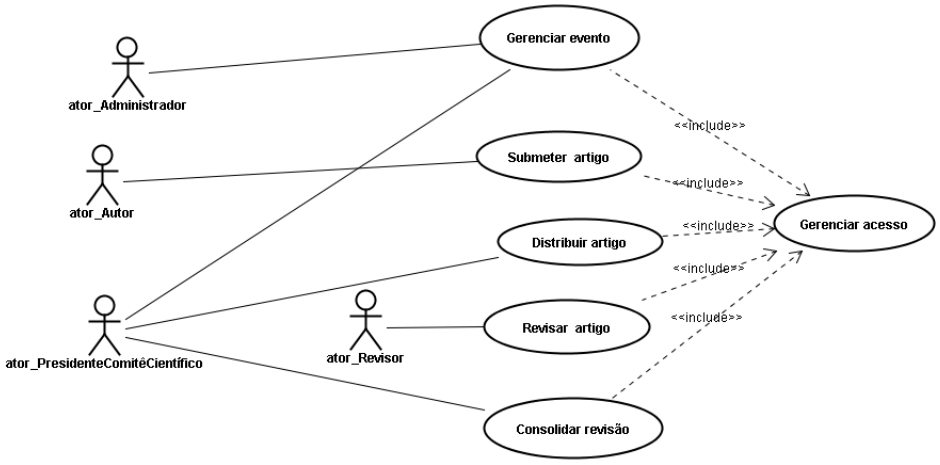

Figura 5.3: Diagrama de Casos de Uso.

5.5 Disciplina Preparação - Teste

Essa disciplina descreve o planejamento das atividades envolvidas no teste do MSPaper, identificando as funcionalidades e características a serem testadas e os responsáveis. A Tabela 5.5 apresenta o Plano de Teste gerado.

Tabela 5.5: Plano de Teste

Identificação do Projeto: MSPaper
Finalidade: Desenvolver e aplicar testes.
Escopo: Este documento descreve o plano de testes ser usado pelo projeto de desenvolvimento da aplicação MSPaper. Fornecendo a todos os membros das equipes envolvidas visibilidade adequada sobre o projeto.
Funcionalidades: Gerenciar evento; Submeter artigo e Revisar artigo.
Restrições: (i) Número de envolvidos no projeto; (ii) Prazo;
Configuração do Teste: *onshore insourcing*
Estratégia de Distribuição: Disciplinas
Sites Envolvidos: Maringá - Gerente Local: Elisa Hatsue Moriya Huzita - Idioma: Português
Papéis e responsabilidades: **Site Maringá:** Planejamento (Teste) - elaborar o plano de teste Preparação (Teste) - refinar o plano de teste Projeto (Teste) - projetar os casos de teste Execução (Teste) - executar os casos de teste
Recursos **Hardware:** (i) Computador pessoal dos participantes; (ii) Servidor para a ferramentade Gerência de Projetos
Cronograma: Data de início: 22/02/2010 Data de término: 20/03/2010

5.6 Disciplina Desenvolvimento - Desenvolvimento

Nessa Disciplina os papéis de Arquiteto e Especificador foram desempenhados pelo mesmo ator. Quanto a comunicação, não foram estabelecidos contatos presenciais. Esses contatos foram realizados utilizando as ferramentas definidas na Disciplina de Planejamento.

A Descrição da Arquitetura para a aplicação MSPaper é apresentada na Figura 5.4.

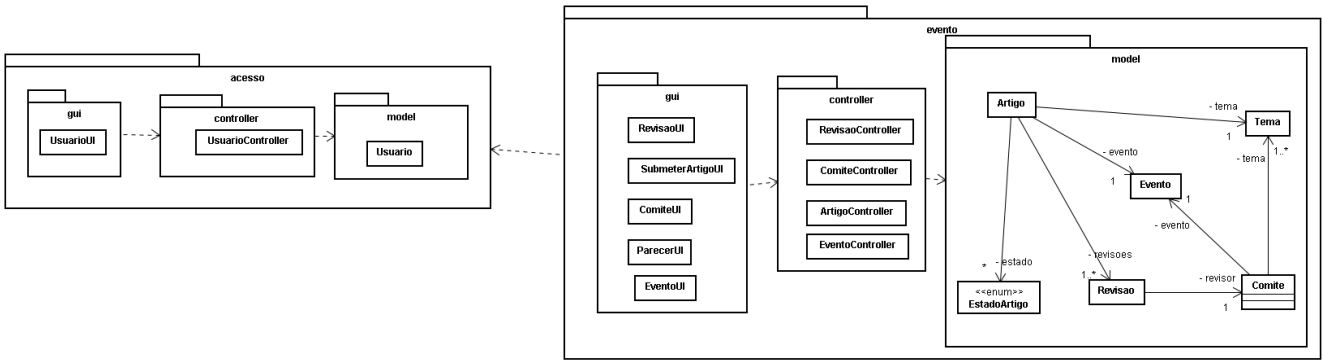

Figura 5.4: Descrição da Arquitetura.

No Diagrama de Classes da Figura 5.5 é possível visualizar a estrutura e a relação entre as classes da aplicação.

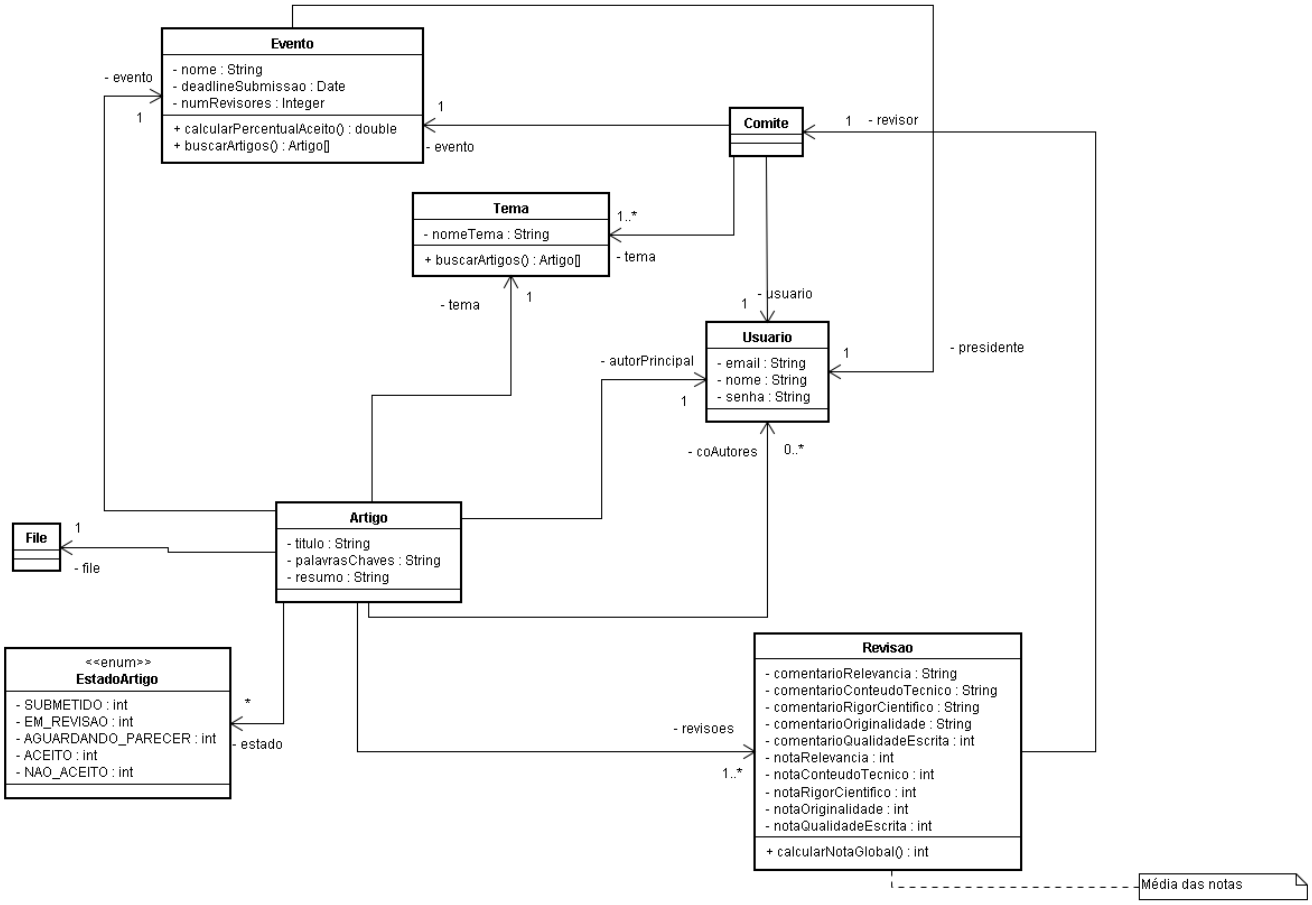

Figura 5.5: Diagrama de Classes.

A Figura 5.6 ilustra o Diagrama de Comunicação - Submissão de Artigos - em que se pode observar a interação entre os objetos por meio das mensagens trocadas. Da mesma forma, foram elaborados diagramas de comunicação para gerenciar evento, distribuir artigo, revisar e consolidar resultados.

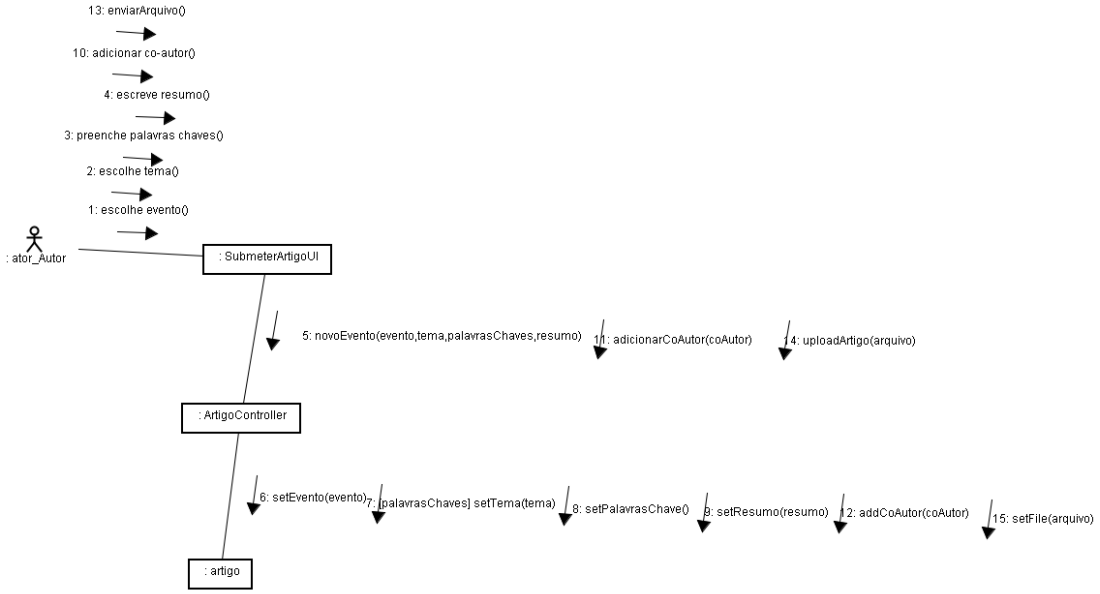

Figura 5.6: Diagrama de Comunicação - Submissão de Artigos.

A Figura 5.7 ilustra o Diagrama de Sequência - Submissão de Artigos - em que se pode observar como os objetos colaboram ao longo do tempo. Do mesmo modo, foram elaborados diagramas de sequência para gerenciar evento, distribuir artigo, revisar e consolidar resultado.

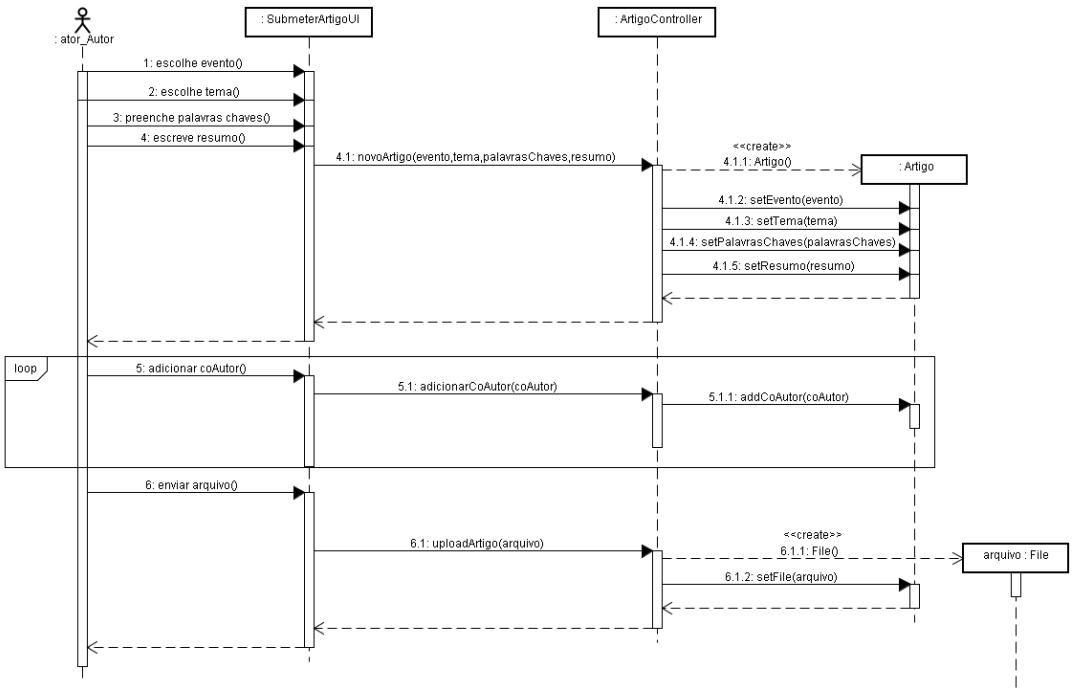

Figura 5.7: Diagrama de Sequência - Submissão de Artigos.

5.7 Disciplina Projeto - Teste

Nessa Disciplina foram detalhados os aspectos técnicos relacionados à condução das atividades de teste. Desse modo, foram gerados três artefatos: Especificação de Projeto de Teste (Tabela 5.6); Especificação de Casos de Teste (Tabela 5.7) e Especificação de Procedimentos de Teste (Tabela 5.8).

Tabela 5.6: Especificação de Projeto de Teste

Identificação do Projeto: MSPaper
Responsável: Site Maringá - Gerente de Projetos: Elisa - Idioma: Português
Técnicas de Teste: Caixa-preta
Critérios de Teste: Descrição do fluxo experado, alternativo e itens requeridos (pré e pós-condições) dos casos de uso e caminhos do diagrama de sequência.
Critérios de Parada: Prazo do projeto.

Tabela 5.7: Especificação de Casos de Teste

Identificação do Projeto: MSPaper			
Responsável: Site Maringá - Gerente de Projetos: Elisa - Idioma: Português			
CT	**Pacote**	**Descrição**	**Dependência**
1	Gerenciar evento	Cadastrar evento e vincular Presidente do Comitê	Usuário
2	Gerenciar evento	Revisões por artigo	-
3	Gerenciar evento	Prazo de submissão	-
4	Gerenciar evento	Data de devolução das submissões	-
5	Gerenciar evento	Data de notificação	-
6	Gerenciar evento	Percentual de artigos aceitos	-
7	Submeter artigo	Enviar artigo	Evento
8	Submeter artigo	Formato do arquivo	-
9	Revisar artigo	Artigos disponíves	Evento Usuário
10	Revisar artigo	Revisão de artigo	-
11	Revisar artigo	Relevância	-
12	Revisar artigo	Conteúdo Técnico	-
13	Revisar artigo	Rigor Científico	-
14	Revisar artigo	Originalidade	-
15	Revisar artigo	Qualidade da Escrita	-
16	Revisar artigo	Avaliação Global	-

Tabela 5.8: Especificação de Procedimentos de Teste

Identificação do Projeto: MSPaper		
Responsável: Site Maringá - Gerente de Projetos: Elisa - Idioma: Português		
CT	**Roteiro**	**Resultado Esperado**
1	1. Inserir evento. 2. Selecionar usuário Presidente Comitê Científico. 3. Definir datas. 4. Adicionar temas. 5. Salvar dados.	Persistência dos dados.
2	1. Inserir número de revisões por artigo menor ou igual a 0. 2. Salvar dados.	Emitir mensagem solicitando valor maior que zero. Não persistir os dados.
3	1. Inserir prazo de submissão menor que a data atual. 2. Salvar dados.	Emitir mensagem informando que o prazo de submissão deve ser posterior a data atual. Não persistir os dados.
4	1. Inserir data de devolução das correções menor que a data atual. 2. Salvar dados.	Emitir mensagem informando que o prazo de devolução deve ser posterior a data de submissão. Não persistir os dados.
5	1. Inserir data de notificação menor que a data atual. 2. Salvar dados.	Emitir mensagem informando que o a data de notificação deve ser posterior a data atual. Não persistir os dados.
6	1. Inserir data de notificação menor que a data de devolução das revisões. 2. Salvar dados.	Emitir mensagem informando que o a data de notificação deve ser posterior a data de devolução das revisões. Não persistir os dados.
7	1. Inserir percentual de artigos aceitos menor ou igual a zero. 2. Salvar dados.	Emitir mensagem informando que o percentual deve ser maior que 0. Não persistir os dados.
8	1. Selecionar o evento. 2. Selecionar tema. 3. Preencher palavras-chave. 4. Inserir resumo. 5. Adicionar coautor. 6. Enviar arquivo formato ".pdf".	Persistência dos dados. *Upload* do arquivo
9	1. Selecionar o evento. 2. Selecionar tema. 3. Preencher palavras-chave. 4. Inserir resumo. 5. Adicionar coautor. 6. Enviar arquivo formato ".doc".	Não persistir os dados. Solicitar aquivo em formato ".pdf".

Tabela 5.9: Continuação da Tabela Especificação de Procedimentos de Teste

10	1. Acessar área de artigos para revisão.	Visualizar artigos dos eventos em que a data atual é menor que a data de devolução da correção.
11	1. Inserir espaços em branco no campo revisão do artigo. 2. Salvar dados.	Emitir mensagem solicitando o preenchimento do campo. Não persistir os dados.
12	1. Inserir espaços em branco no campo relevância. 2. Salvar dados.	Emitir mensagem solicitando o preenchimento do campo. Não persistir os dados.
13	1. Inserir espaços em branco no campo conteúdo técnico. 2. Salvar dados.	Emitir mensagem solicitando o preenchimento do campo. Não persistir os dados.
14	1. Inserir espaços em branco no campo rigor científico. 2. Salvar dados.	Emitir mensagem solicitando o preenchimento do campo. Não persistir os dados.
15	1. Inserir espaços em branco no campo originalidade. 2. Salvar dados.	Emitir mensagem solicitando o preenchimento do campo. Não persistir os dados.
16	1. Inserir espaços em branco no campo qualidade da escrita. 2. Salvar dados.	Emitir mensagem solicitando o preenchimento do campo. Não persistir os dados.
17	1. Inserir espaços em branco no campo avaliação global. 2. Salvar dados.	Emitir mensagem solicitando o preenchimento do campo. Não persistir os dados.

5.8 Disciplina Implementação - Desenvolvimento

Essa Disciplina foi executada pela equipe de Belo Horizonte-MG, que tomando como base os artefatos gerados nas Disciplinas anteriores e utilizando o ambiente de desenvolvimento Netbeans 6.8 e o sistema de gerenciamento de banco de dados Postgresql, a aplicação MSPaper foi desenvolvida.

A Figura 5.8 mostra a interface de login para acesso ao MSPaper.

Figura 5.8: Interface de Login.

Na Figura 5.9 tem-se a interface em que o Presidente do Comitê Científico pode visualizar os eventos nos quais é *chair*.

Figura 5.9: Interface da Lista de Eventos.

A interface para submissão de artigos é ilustrada na Figura 5.10.

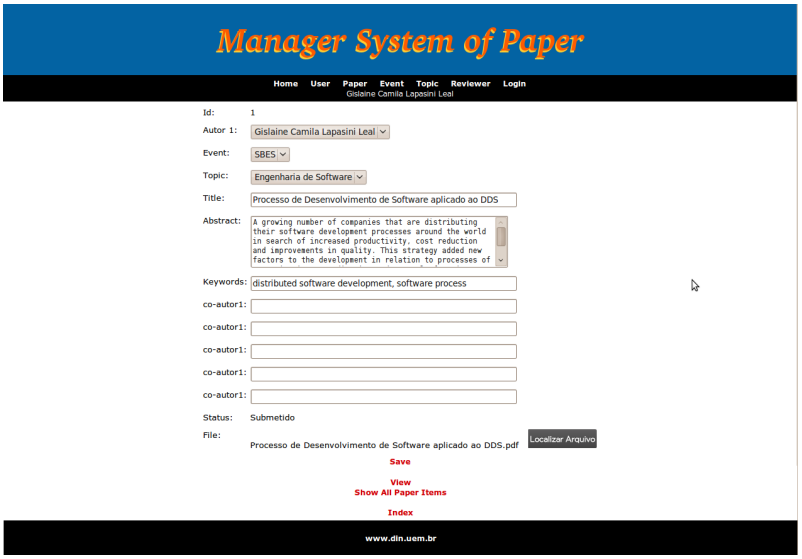

Figura 5.10: Interface de Submissão de Artigo.

As Figuras 5.11 e 5.12 mostram a lista de artigos que o revisor possui para revisar e a interface em que os dados da revisão são registrados, respectivamente.

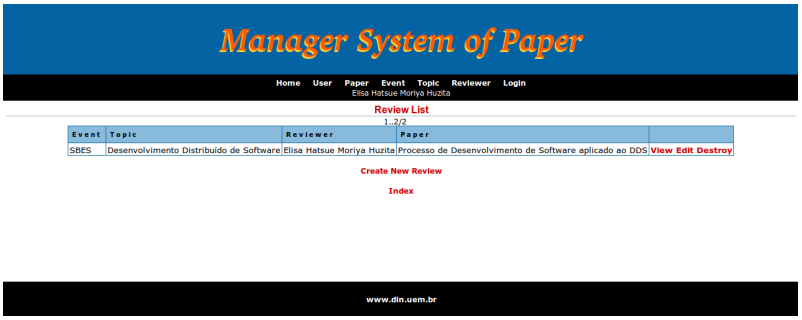

Figura 5.11: Interface do Revisor de Artigo.

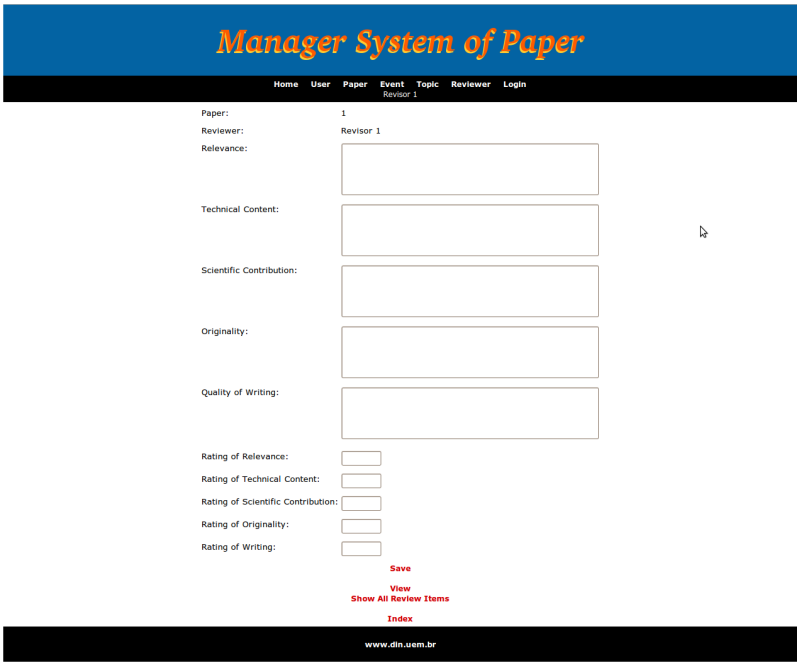

Figura 5.12: Interface de Revisão.

5.9 Disciplina Execução - Teste

Nessa disciplina foram executadas e registradas as atividades de teste. Devido ao escopo reduzido do projeto não houveram incidentes de teste.

Os artefatos Diário de Testes e Resumo de Teste são apresentados nas Tabelas 5.10 e 5.11, respectivamente.

Tabela 5.10: Diário de Teste

Identificação do Projeto: MSPaper				
Responsável: Site Maringá - Gerente de Projetos: Elisa - Idioma: Português				
Ambiente de execução: Intel Core 2 Duo Processor T6400, 4GB SDRAM, 320GB HDD				
CT	**Responsável**	**Resultado**	**Eventos não Esperado**	**Data**
1	Camila	Dados persistidos corretamente.	-	22/03/2010
2	Camila	Mensagem solicitando valor maior que 0.	-	22/03/2010
3	Camila	Mensagem informando que a data de submissão deve ser maior que a data atual.	-	22/03/2010
4	Camila	Mensagem informando que a data de devolução deve ser maior que a data de submissão.	-	22/03/2010
5	Camila	Mensagem informando que a data de notificação deve ser maior que a data atual.	-	22/03/2010
6	Camila	Mensagem informando que a data de notificação deve ser posterior a data de devolução das revisões.	-	22/03/2010
7	Camila	Dados persistidos.	Percentual menor que zero.	22/03/2010
8	Camila	Dados persistidos e arquivo carregado	-	22/03/2010
9	Camila	Dados persistidos.	*Upload* formato ".doc"	22/03/2010
10	Camila	Visualização dos artigos que a data de atual é menor que a data de devolução da correção.		22/03/2010
11	Camila	Dados não persistidos.	-	22/03/2010
12	Camila	Dados não persistidos	-	22/03/2010
13	Camila	Dados não persistidos	-	22/03/2010
14	Camila	Dados não persistidos	-	22/03/2010
15	Camila	Dados não persistidos	-	22/03/2010
16	Camila	Dados não persistidos	-	22/03/2010
17	Camila	Dados não persistidos	-	22/03/2010

Tabela 5.11: Resumo do Teste

Identificação do Projeto: MSPaper	
Responsável: Site Maringá - Gerente de Projetos: Elisa - Idioma: Português	
Início dos Testes: 21/03/2010 ‖ **Fim dos Testes:** 22/03/2010	
Números do Teste	
Casos de Teste criados antes do início dos testes	16
Casos de Teste criados durante os testes	1
Casos de Teste executados	17
Casos de Teste executados com sucesso	15
Casos de Teste executados com erro	2
Casos de Teste enviados para correção	2
Percentuais do Teste	
Casos de Teste executados	100%
Casos de Teste executados com sucesso	88%
Casos de Teste executados com incidência de erro	0%
Casos de Teste corrigidos	100%

5.10 Considerações Finais

A prova de conceito apresentada ilustra como a abordagem descrita no Capítulo 4 pode ser instanciada, não tendo como objetivo extrair conclusões sobre a real viabilidade e funcionalidade da abordagem. Algumas dificuldades encontradas e que inviabilizaram a condução de um estudo experimental foram:

- no âmbito regional, há dificuldade em encontrar empresas que utilizem essa configuração de desenvolvimento.
- os projetos desenvolvidos utilizando a estratégia distribuída, geralmente, são de grande porte, o que demandaria um prazo maior para o acompanhamento do projeto.
- dificuldade de reunir equipes dispersas, mesmo no meio acadêmico, que tenham disponibilidade para participar do experimento.

A prova de conceito exposta apresenta limitações em relação ao número de participantes e equipes envolvidas. Além disso, as equipes utilizam o mesmo idioma, as diferenças socioculturais são pequenas e não há diferença de fuso horário. No entanto, esta prova de conceito não tem o intuito de validar a abordagem proposta mas, sim elucidar o seu funcionamento. O Capítulo seguinte descreve o estudo de viabilidade realizado.

Capítulo

6

Estudo de Viabilidade

Este capítulo apresenta um estudo de viabilidade, também denominado de projetos quasi-experimental, da abordagem descrita no Capítulo 4. Esse estudo tem por objetivo coletar evidências, pontos fortes e fracos sobre a abordagem proposta.

A principal característica de um estudo de viabilidade é que os dados são coletados de acordo com algum projeto experimental, mas não há controle sobre todas as variáveis. O objetivo não é encontrar uma resposta definitiva, mas sim, construir um corpo de conhecimento que aborda a plausibilidade da continuidade do estudo, gerar nova hipóteses sobre a abordagem e sua utilidade (Shull et al., 2001).

6.1 Definição dos Objetivos

Esta etapa discorre sobre os objetivos do estudo de viabilidade sobre três perspectivas: global, da medição e do estudo.

6.1.1 Objetivo Global

Caracterizar a viabilidade da abordagem integrada de desenvolvimento e teste de software para equipes distribuídas.

6.1.2 Objetivo da Medição

Investigar e caracterizar a abordagem com relação à viabilidade para o contexto de DDS.

6.1.3 Objetivo do Estudo

Analisar a abordagem integrada de desenvolvimento e teste de software para equipes distribuídas.

Com o propósito de caracterizar

Com respeito à viabilidade das atividades, artefatos e papéis definidos pela abordagem

Do ponto de vista do pesquisador

No contexto de pessoas envolvidas com o desenvolvimento distribuído de software.

6.1.4 Questões

- Q1: As atividades presentes na abordagem são suficientes para apoiar o DDS?
  Métrica: A lista de atividades oferecidas pela abordagem.

- Q2: O artefatos presentes na abordagem são suficientes para apoiar o DDS, servindo como comunicação entre as equipes de desenvolvimento e teste?
  Métrica: A lista de artefatos especificados pela abordagem.

6.2 Planejamento

Esta etapa prepara como o estudo será conduzido e compõe-se pelos seguintes passos: formulação da hipótese, seleção do contexto, seleção de variáveis, seleção de participantes e projeto do estudo.

6.2.1 Definição das Hipóteses

- H0: A abordagem integrada de desenvolvimento e teste de software não contempla todas as peculiaridades do desenvolvimento distribuído.

- H1: A abordagem integrada de desenvolvimento e teste de software contempla todas as peculiaridades do desenvolvimento distribuído.

6.2.2 Seleção do Contexto

O estudo foi conduzido por meio de questionários que foram enviados eletronicamente aos participantes com conhecimento no domínio que abrange o estudo.

6.2.3 Seleção dos Participantes

Os participantes foram selecionados com base nos conhecimentos sobre desenvolvimento distribuído de software e disponibilidade, ou seja, não foram escolhidos de forma aleatória.

O estudo foi conduzido no ambiente acadêmico, que de acordo com (Shull et al., 2001) é adequado para estudos de viabilidade. Embora os resultados não possam ser aplicados diretamente no meio industrial, permitem que novos conceitos sejam testados antes de seu uso na indústria.

6.2.4 Projeto do Estudo

Após a tabulação dos dados, será realizada uma análise descritiva dos resultados e medidas de dispersão. Em seguida será aplicado o teste qui-quadrado para analisar os resultados. Esse teste se destina a encontrar um valor de dispersão para duas variáveis nominais, avaliando a associação existente entre variáveis qualitativas. O princípio básico deste método é comparar proporções, isto é, as possíveis divergências entre as freqüências observadas e esperadas para um dado evento.

A escala de mensuração adotada para as variáveis dependentes, relacionadas ao perfil do entrevistado: 0 - Nenhum; 1 - Básico; 2 - Intermediário; 3 - Avançado.

A escala de mensuração utilizada para as variáveis independentes, relacionadas ao estudo de viabilidade: 0 - Nenhum; 1 - Baixo; 2 - Intermediário; 3 - Satisfatório.

6.2.5 Instrumentação

Com o objetivo de caracterizar cada participante foi utilizado o questionário Perfil do Entrevistado, contendo questões que estão relacionadas com as variáveis independentes, por exemplo: experiência com desenvolvimento de sistemas, conhecimento em desenvolvimento distribuído de software, gerência de projetos e teste de software.

Com o objetivo de avaliar a abordagem integrada foi utilizado o questionário Estudo de Viabilidade, contendo as respostas, que correspondem às variáveis dependentes de cada participante, por exemplo: grau em que o conjunto de disciplinas, atividades, artefatos atende ao DDS. Foi elaborado um memorial descritivo apresentando o objetivo do estudo, as principais caracaterísticas da abordagem e o Diagrama de Pacotes de cada uma das disciplinas, em que é possível visualizar os papéis, atividades e artefatos envolvidos.

Os documentos utilizados na instrumentação podem ser vistos no Apêndice E.

6.2.6 Validade

Uma questão fundamental com relação aos resultados do estudo é identificar quão válidos eles são, pois estes devem ser generalizados. Há quatro tipos de validade: interna, externa, construtiva e conclusiva, as quais são descritas a seguir (Travassos, 2002).

Validade Interna

Esse tipo de validade refere-se ao tratamento-resultado e seus riscos estão relacionados aos participantes, que podem ficar cansados ou desanimados. Desse modo, reforça-se a necessidade de uma instrumentação preparada adequadamente para evitar mal entedimento, o que pode afetar os resultados negativamente (Travassos, 2002).

No estudo em questão os participantes foram selecionados com base no conhecimento sobre desenvolvimento distribuído de software. A validade interna é dependente do número de participantes, quanto maior o número de participantes melhor será a validade interna.

Validade Externa

A validade externa é relacionada às condições que limitam a habilidade de generalizar os resultados. Essa validade é dependente da capacidade do estudo refletir o mesmo comportamento em outros grupos de participantes e profissionais da indústria, ou seja, em outros grupos além daquele em que o estudo foi aplicado (Travassos, 2002).

Os dados do perfil dos indivíduos podem ser analisados para avaliar o nível de experiência e conhecimento em desenvolvimento distribuído de software. O maior problema que pode ocorrer em relação à validade externa é elevado número de pessoas do meio acadêmico em relação ao meio industrial.

Validade Construtiva

A validade construtiva considera os relacionamentos entre a teoria e a observação, ou seja, se o tratamento reflete bem a causa e o resultado reflete bem o efeito. Os problemas a serem considerados referem-se ao fato dos participantes se basearem nas hipóteses e na expectativa do pesquisador (Travassos, 2002).

Validade Conclusiva

Esse tipo de validade relaciona-se à habilidade de chegar a uma conclusão correta a respeito dos relacionamentos entre o tratamento e os resultados do experimento. Para

tanto são considerados os seguintes elementos: teste estatístico e o tamanho do conjunto de participantes (Travassos, 2002).

A validade conclusiva mensura a relação entre o tratamento e o resultado, determinando a capacidade do estudo em gerar alguma conclusão. A obtenção de boas conclusões será permitida por meio de uma boa definição das variáveis independentes e dependentes juntamente com as análises descritivas adequadas e o teste qui-quadrado.

6.3 Operação

Um roteiro (memorial descritivo) explicando a abordagem, juntamente com os questionários do perfil do entrevistado e estudo de viabilidade foram encaminhados para 16 pessoas.

O roteiro explica a finalidade do estudo, os objetivos da abordagem e apresenta o Diagrama de Pacotes de cada Disciplina, que contém as atividades, os papéis envolvidos e artefatos gerados.

6.3.1 Resultados do Estudo

A Tabela 6.1 apresenta os resultados obtidos para as variáveis independentes (A - C) e dependentes (D- L), em que:

- **(A)**: Conhecimento em Desenvolvimento Distribuído de Software.
- **(B)**: Experiência em Gerência de Projetos.
- **(C)**: Conhecimento em teste de software.
- **(D)**: Conjunto de disciplinas é satisfatório para o DDS.
- **(E)**: Conjunto de atividades atende às necessidades do DDS.
- **(F)**: Conjunto de artefatos atende às necessidades do DDS.
- **(G)**: Uso de notação (UML) oferece informações relevantes para as duas equipes (desenvolvimento e teste) pode amenizar os problemas de comunicação.
- **(H)**: Identificação das equipes, bem como de seus membros e responsabilidades através dos Planos de Desenvolvimento Global, Plano de Desenvolvimento Local e Plano de Testes.

- **(I)**: Nomenclatura comum a todos os envolvidos no projeto possibilita entender a dependência das atividades e artefatos.
- **(J)**: Uso de OCL (*Object Constraint Language*) para especificar restrições.
- **(K)**: Impacto da padronização dos artefatos e atividades na qualidade dos produtos desenvolvidos.
- **(L)**: Atividades de teste ao longo de todas as disciplinas.

Na Tabela 6.1 a escala de mensuração adotada é: 0 - Nenhum; 1 - Baixo; 2 - Intermediário; 3 - Satisfatório.

Tabela 6.1: Resultados do Estudo de Viabilidade

N	A	B	C	D	E	F	G	H	I	J	K	L
1	2	0	1	3	3	3	2	2	3	2	3	3
2	2	2	2	3	3	3	2	2	3	3	3	3
3	2	1	2	3	3	3	2	3	2	2	2	3
4	2	1	1	2	2	2	3	2	3	3	2	3
5	2	0	1	3	3	2	2	3	3	2	2	3
6	3	1	1	3	2	3	2	3	2	2	3	3
7	2	2	2	2	2	3	3	2	3	3	3	3
8	2	1	1	2	2	3	3	2	2	3	2	2
9	3	2	2	3	3	3	3	3	2	2	1	3
10	1	1	1	2	3	3	3	3	2	2	3	3
11	3	1	1	3	3	3	3	2	3	2	2	3
12	2	0	3	3	3	3	3	3	3	3	3	2

6.4 Análise e Interpretação dos Dados

A Análise e Interpretação dos Dados divide-se em quatro etapas: Validação dos Dados, Estatística Descritiva e Análise, Aplicação do Teste Estatístico e Verificação das Hipóteses. A seguir essas etapas são descritas.

6.4.1 Validação dos Dados

No estudo foram utilizados 12 participantes, todos responderam ao questionário Perfil do Entrevistado e Estudo de Viabilidade. Não foram encontrados *outliers* nas respostas dos questionários.

6.4.2 Estatística Descritiva e Análise

A tabulação e apresentação de dados são fundamentais ao bom julgamento estatístico pois permitem focar em características relevantes a serem utilizadas na solução de problemas. Desse modo, as medidas de tendência central, no caso mediana e moda, pois os valores estão em escala ordinal, organizam a amostra destacando os acontecimentos (Montgomery e Runger, 2009).

A mediana divide os dados em duas partes iguais. Se o número de observações for par, a mediana estará na metade da distância entre os dois valores centrais. Se o número de observações for ímpar, a mediana será o valor central. A moda é o valor da observação que ocorre com mais freqüência (Montgomery e Runger, 2009).

A Tabela 6.2 apresenta a estatística descritiva (mediana e a moda) dos dados coletados.

Tabela 6.2: Estatística Descritiva

	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**
Mediana	2	1	1	3	3	3	3	2,5	3	2	2,5	3
Moda	2	1	1	3	3	3	3	2	3	2	3	3

Após a tabulação dos dados foram gerados gráficos utilizando a ferramenta GNUPlot 4.0, que são apresentados nas Figuras 6.1, 6.2, 6.3, 6.4, 6.5, 6.6, 6.7, 6.8 e 6.9. Esses gráficos ilustram a relação entre o conhecimento dos participantes em Desenvolvimento Distribuído de Software, Experiência em Gerência de Projetos/Conhecimento em teste de software, e, a terceira coordenada, representa as características da abordagem proposta que foram analisadas: Disciplinas (Figura 6.1), Atividades (Figura 6.2), Artefatos (Figura 6.3, uso de UML para amenizar os problemas de comunicação (6.4), Identificação das Equipes para aumentar a percepção (Figura 6.5), uso de OCL para amenizar a imprecisão dos artefatos (Figura 6.7), padronização dos artefatos e atividades (Figura 6.8) e atividades de teste (Figura 6.9).

Nas Figura 6.1, 6.2 e 6.3 pode-se observar que os participantes com conhecimento intermediário/avançado em DDS, experiência básica/intermediária em Gerência de Projetos avaliaram que as Disciplinas, Atividades e Artefatos propostos na abordagem contemplam de modo intermediário/satisfatório às necessidades do DDS. O vale apresentado representa os participantes que não possuem experiência em Gerência de Projetos.

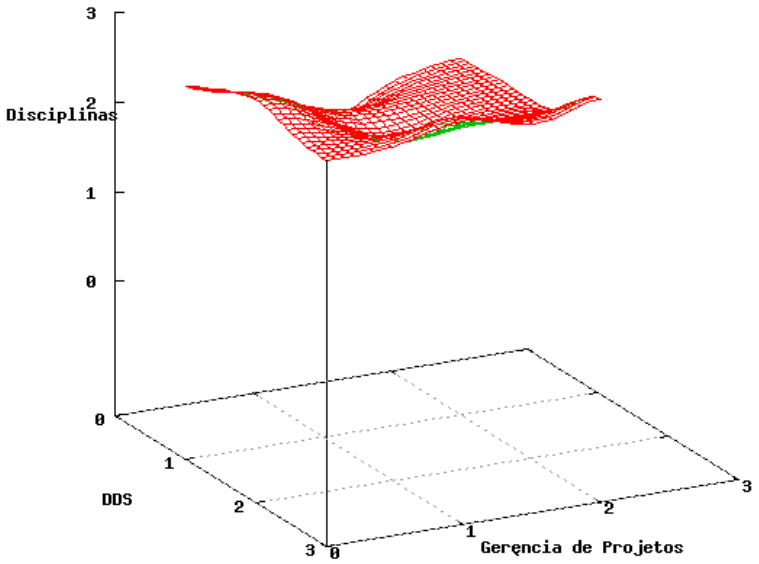

Figura 6.1: Gráfico da Relação Conhecimento em DDS x Experiência em Gerência de Projetos x Disciplinas.

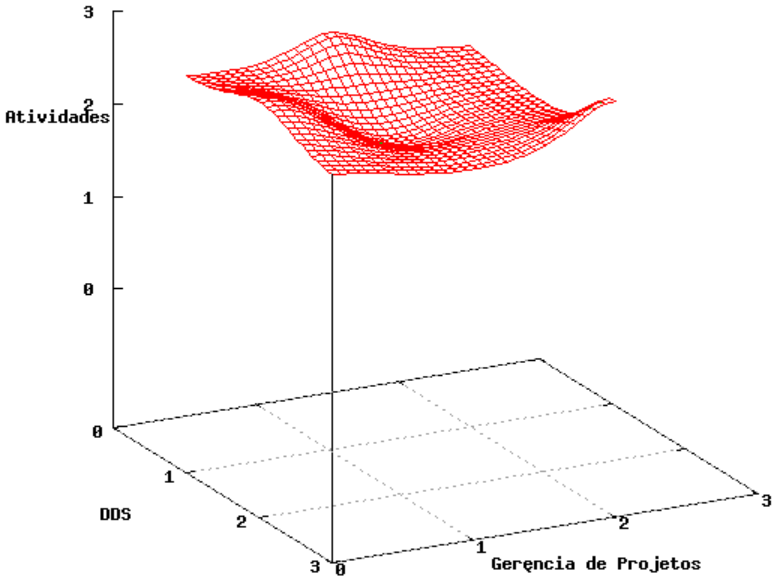

Figura 6.2: Gráfico da Relação Conhecimento em DDS x Experiência em Gerência de Projetos x Atividades.

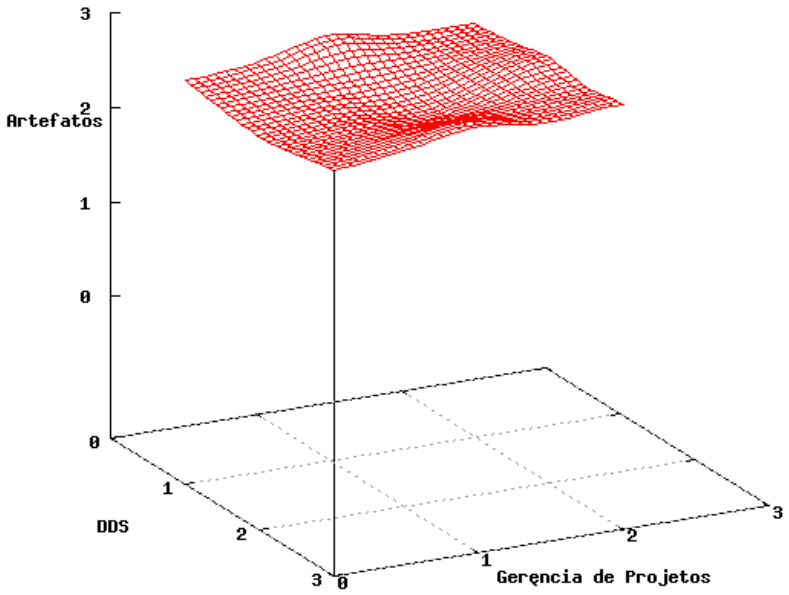

Figura 6.3: Gráfico da Relação Conhecimento em DDS x Experiência em Gerência de Projetos x Artefatos.

A Figura 6.4 mostra que os participantes com conhecimento intermediário/avançado em DDS, indiferente da experiência em Gerência de Projetos avaliaram que a notação UML, através das informações oferecidas para as equipes de desenvolvimento e teste, pode amenizar os problemas de comunicação em grau intermediário/satisfatório.

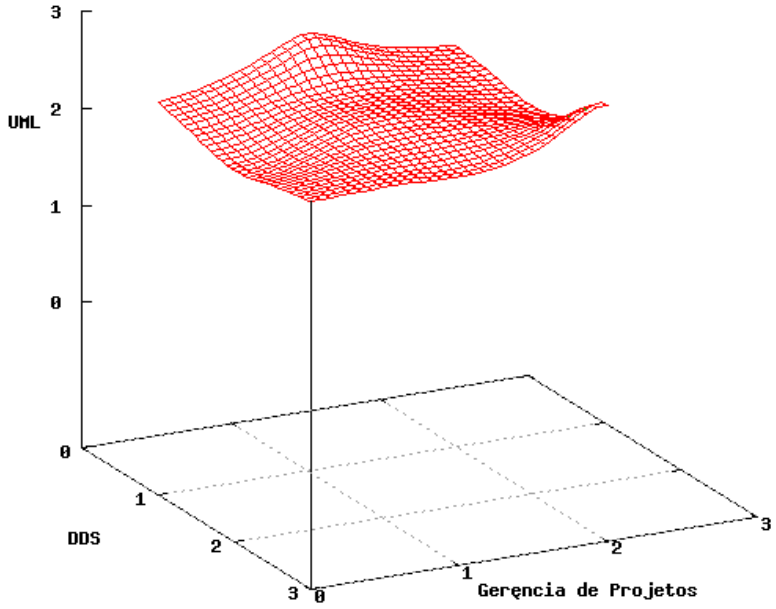

Figura 6.4: Gráfico da Relação Conhecimento em DDS x Experiência em Gerência de Projetos x UML.

Na Figura 6.5, o gráfico evidencia dois picos, em relação a identificação das equipes e seus membros como mecanismo para aumentar a percepção de modo intermediário/satisfatório.

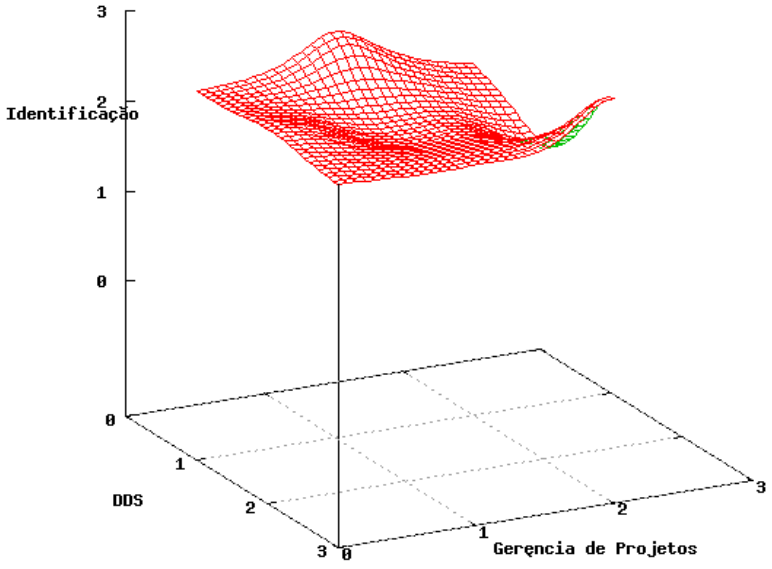

Figura 6.5: Gráfico da Relação Conhecimento em DDS x Experiência em Gerência de Projetos x Identificação das Equipes.

Na Figura 6.6, é possível evidenciar um pico e um vale, os quais são dados pela variação da experiência em gerência de projetos dos participantes. No entanto, observa-se que a abordagem atende de modo intermediário/satisfatório as necessidades do DDS por oferecer uma nomenclatura comum a todos os envolvidos no projeto e possibilitar o entendimento das dependências entre as atividades e artefatos.

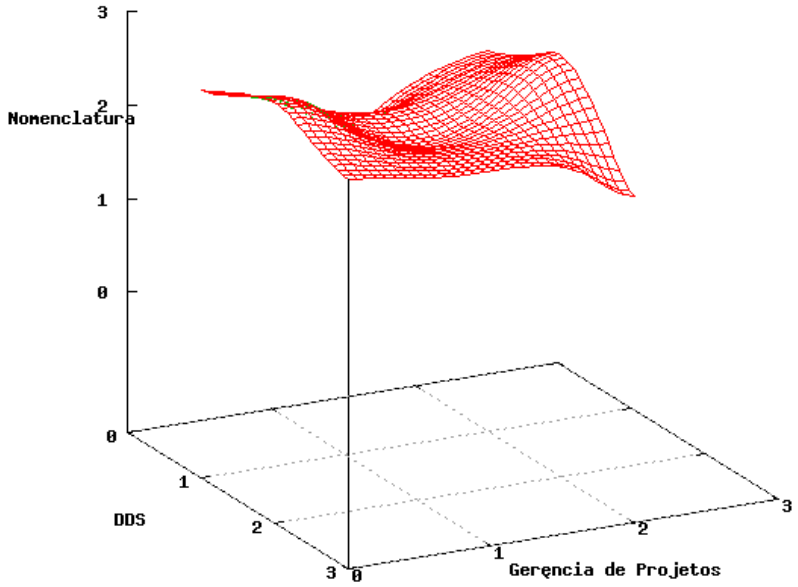

Figura 6.6: Gráfico da Relação Conhecimento em DDS x Experiência em Gerência de Projetos x Nomenclatura.

O gráfico da Figura 6.7 apresenta dois picos, os quais são formados devido a variação da experiência em gerência de projetos. Mas, de modo geral a OCL foi avaliada como sendo uma solução intermediária/satisfatória para os problemas de imprecisão dos artefatos.

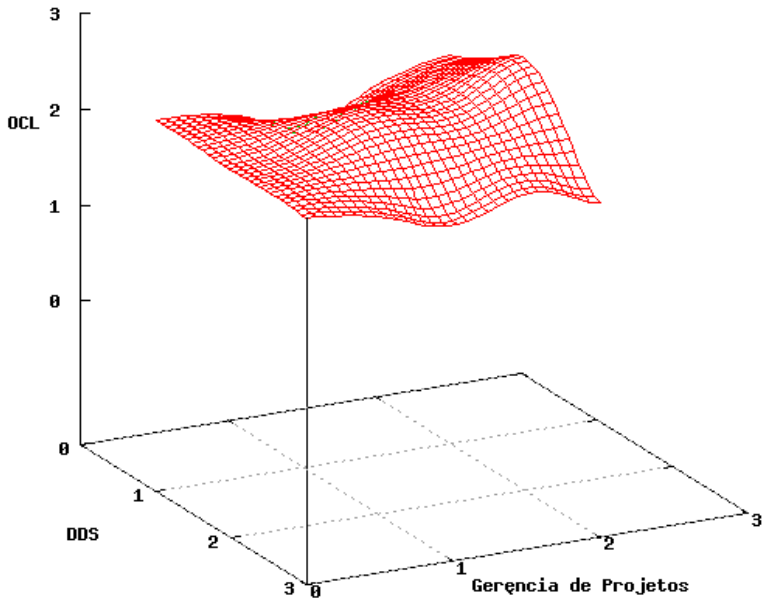

Figura 6.7: Gráfico da Relação Conhecimento em DDS x Experiência em Gerência de Projetos x OCL.

Na Figura 6.8, o gráfico evidencia um vale e uma depressão. O vale representa os participantes com conhecimento básico/intermediário em DDS e nenhuma/básica experiência em gerência de projetos que avaliaram como intermediário o grau em que a padronização dos artefatos e atividades podem impactar na qualidade dos produtos desenvolvidos. A depressão demonstra que participantes com conhecimento intermediário em DDS e experiência intermediária avaliaram que a padronização não é determinante na qualidade dos produtos.

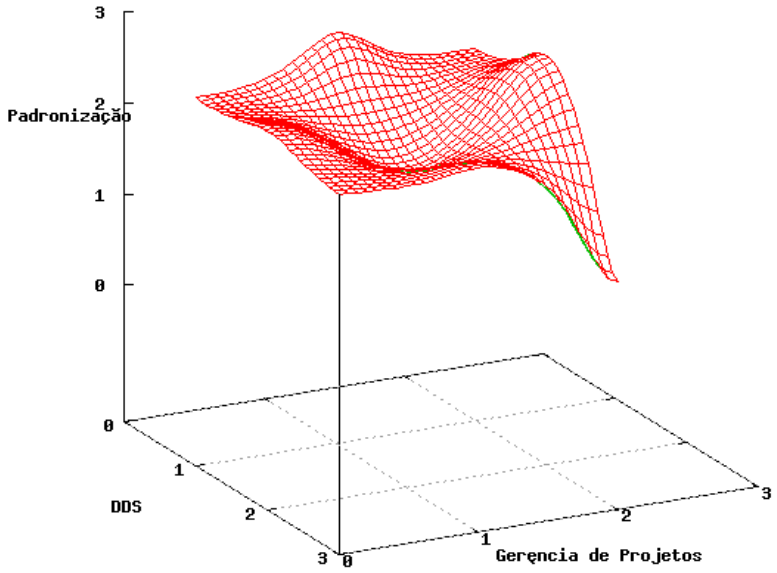

Figura 6.8: Gráfico da Relação Conhecimento em DDS x Experiência em Gerência de Projetos x Padronização.

O gráfico da Figura 6.9 apresenta uma depressão, que representa os participantes com conhecimento avançado em teste de software e conhecimento intermediário em DDS que avaliaram que as atividades de teste ao longo de todas as disciplinas de desenvolvimento podem reduzir os problemas de integração de modo intermediário. Já os participantes com conhecimento básico/intermediário em teste analisaram como sendo satisfatório a redução dos problemas de integração.

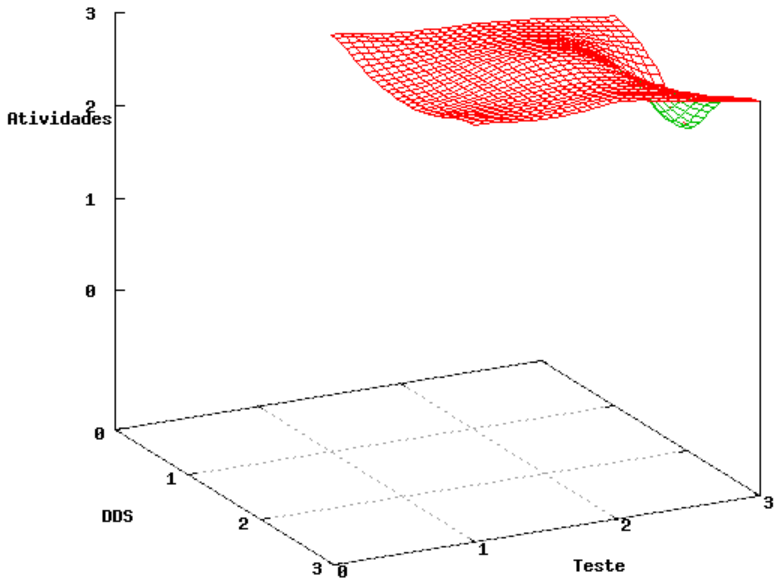

Figura 6.9: Gráfico da Relação Conhecimento em DDS x Conhecimento em Teste x Atividades de Teste ao longo das Disciplinas.

6.4.3 Aplicação do Teste Estatístico

Os participantes foram classificados em apenas um nível de conhecimento à partir das variáveis de entrada, conforme pode ser visto na Tabela 6.3).

Tabela 6.3: Classificação do Nível de Conhecimento do Participante

Participante	Nível de Conhecimento
1	1
2	2
3	2
4	1
5	1
6	1
7	2
8	1
9	2
10	1
11	1
12	1

As Variáveis Dependentes foram relacionadas com a Variável Independente que mede o conhecimento do participante através do teste estatístico Qui-Quadrado. Como o teste Qui-Quadrado considera a comparação das distribuições, podemos comparar a distribuição de respostas dos participantes com a distribuição ideal.

A função densidade de probabilidade qui-quadrado é representada por x_v^2, em que v designa o número de graus de liberdade da x^2. Se v observações de uma variável são independentes, então o número de graus de liberdade é igual a v. Entretanto, um grau de liberdade é perdido para cada restrição sobre as v observações (Montgomery e Runger, 2009).

A função distribuição é dada por $P(X^2 >= X_v^2) = \alpha$, em que α é a probabilidade de somas dos quadrados iguais ou superiores ao valor correspondente tabelado. O nível de significância α é, geralmente, fixado em torno do valor 0,05. Quanto maior α , maior é o risco de rejeitar hipóteses boas; inversamente, o risco de aceitar hipóteses falsas aumenta, na medida que o valor de α diminue. Se o valor calculado da variável aleatória x^2 for maior do que o valor tabelado x_v^2, rejeitase a hipótese de que as variáveis sejam aleatórias, entretanto, se ele for menor ou igual, a hipótese é aceita (Montgomery e Runger, 2009).

A Tabela 6.4 mostra o nível esperado em que a abordagem contempla às necessidades do DDS segundo o conhecimento dos participantes.

Tabela 6.4: Nível Esperado que a abordagem contempla às necessidades do DDS.

	Nível Esperado			
Nível de conhecimento	**0**	**1**	**2**	**3**
0				
1		8		
2			4	
3				

As Tabelas 6.5, 6.6 6.7, 6.8, 6.9, 6.10, 6.11, 6.12 e 6.13 apresentam a distribuição dos dados de acordo o nível de conhecimento dos participantes.

Tabela 6.5: Distribuição do Grau que o Conjunto de Disciplinas atende às necessidades do DDS.

	Disciplinas			
Nível de conhecimento	**0**	**1**	**2**	**3**
0				
1			3	5
2			1	3
3				

O α adotado é de 0,05 e v é 9. Desse modo, o X_v^2 calculado para os dados da Tabela 6.5 foi 6,125.

Tabela 6.6: Distribuição do Grau que o Conjunto de Atividades atende às necessidades do DDS.

	Atividades			
Nível de conhecimento	**0**	**1**	**2**	**3**
0				
1			3	5
2			1	3
3				

O X_v^2 obtido para os dados da Tabela 6.6 foi 6,125.

Tabela 6.7: Distribuição do Grau que o Conjunto de Artefatos atende às necessidades do DDS.

	Artefatos			
Nível de conhecimento	**0**	**1**	**2**	**3**
0				
1			2	6
2				4
3				

O X_v^2 obtido para os dados da Tabela 6.7 foi 0,04847.

Tabela 6.8: Distribuição do Grau que a Notação UML pode amenizar problemas de comunicação.

	Notação UML			
Nível de conhecimento	**0**	**1**	**2**	**3**
0				
1			3	5
2			2	2
3				

Para os dados das Tabelas 6.8 e 6.9 o X_v^2 calculado foi 4,5.

Tabela 6.9: Distribuição do Grau que a Identificação das Equipes pode aumentar a percepção.

	Identificação das Equipes			
Nível de conhecimento	**0**	**1**	**2**	**3**
0				
1			4	4
2			2	2
3				

Tabela 6.10: Distribuição do Grau que a Abordagem Atende às necessidades do DDS.

	Nomenclatura			
Nível de conhecimento	**0**	**1**	**2**	**3**
0				
1			3	5
2			2	2
3				

O X_v^2 obtido para os dados da Tabela 6.10 foi 3,125.

Tabela 6.11: Distribuição do Grau que a OCL pode amenizar a ambiguidade.

	OCL			
Nível de conhecimento	**0**	**1**	**2**	**3**
0				
1			5	3
2			2	2
3				

O X_v^2 obtido para os dados da Tabela 6.11 foi 4,5.

Tabela 6.12: Distribuição do Grau que a Padronização de atividades e artefatos pode impactar na qualidade.

	Padronização			
Nível de conhecimento	**0**	**1**	**2**	**3**
0				
1			4	4
2		1	1	2
3				

O X_v^2 calculado para os dados da Tabela 6.12 foi 6,125.

Tabela 6.13: Distribuição do Grau em que as atividades de teste ao longo das disciplinas podem reduzir os problemas de integração.

	Atividades de Teste			
Nível de conhecimento	**0**	**1**	**2**	**3**
0				
1			2	6
2				4
3				

Para os dados da Tabela 6.13 o X_v^2 calculado foi 0,04847.

6.4.4 Verificação das Hipóteses

Pela tabela da distribuição do qui-quadrado $X_v^2 = 7{,}815$. Ao aplicar o teste qui-quadrado todas as tabelas relacionadas as variáveis dependentes tiveram como resposta $X^2 >= X_v^2$. Desse modo, rejeita-se H1 em prol de H0. Com isso, podemos dizer que *"A abordagem integrada de desenvolvimento e teste de software não contempla todas as peculiaridades do desenvolvimento distribuído"*.

Como a amostra do estudo foi pequena tem-se os seguintes problemas relacionados à validade:(1) A potência do teste estatístico qui-quadrado é baixa, o que afeta a validade de conclusão; (2) Como o nível de conhecimento dos participantes é próximo, não há diversidade dos perfis, o que compromete a validade externa (generalização dos resultados).

Além disso, observa-se que o estudo pode ter sido influenciado pela classificação do conhecimento em apenas um nível pois, a maioria dos participantes possuia conhecimento intermediário ou avançado em DDS, mas apresentava nenhuma ou pouca experiência em gerência de projetos e com conhecimento básico em teste de software. Assim, é possível estratificar a análise utilizando cada um dos tipos de conhecimento do participante.

6.5 Considerações Finais

O teste Qui-Quadrado foi comprometido na generalização dos resultados devido ao reduzido número de participantes, ou seja, para a aplicação de um teste estatístico, deveria existir uma amostra maior do que a que foi utilizada. No entanto, a partir da análise dos gráficos apresentados na Seção 6.4.2, que correlacionam os conhecimentos do participante com as características analisadas da abordagem, pode-se evidenciar que as peculiaridades do DDS são contempladas em um nível intermediário/satisfatório.

A condução do estudo de viabilidade possibilitou levantar, junto aos participantes, os seguintes pontos:

- inserir atividades que envolvam cooperação e percepção das equipes distruibuídas que estejam envolvidas em atividades conjuntas;
- englobar atividades de revisão das atividades e artefatos, o que afeta diretamente a qualidade; e,
- gerar um artefato de conclusão de disciplina.

Capítulo

7

Conclusões

A crescente busca por maior competitividade tem levado as empresas a adotarem o DDS. Tentando realizar desenvolvimento a baixo custo, empresas têm atravessado fronteiras, formando um mercado global. A produção distribuída pode ser motivada, também, por questões de qualidade, gerenciamento do controle de produção, domínio de tecnologias empregadas, redução da necessidade de grandes contratações imediatas ou para transferir conhecimento a uma filial (L'Erário, 2009b).

Essa mudança de paradigma tem causado impacto no marketing, na distribuição e na forma de concepção, de produção, de projeto, de teste e de entrega do software aos clientes (Sangwan et al., 2006). Segundo Damian e Lanubile (2004) para minimizar esses efeitos e alcançar níveis mais elevados de produtividade são necessárias novas tecnologias, processos e métodos compatíveis com a abordagem de desenvolvimento distribuído.

Diante deste cenário o objetivo deste trabalho foi apresentar uma abordagem integrada de desenvolvimento e teste de software para apoiar o desenvolvimento com equipes distribuídas. A abordagem foi desenvolvida com o intuito de oferecer suporte adequado aos processos de comunicação, coordenação e controle fornecendo uma nomenclatura comum aos envolvidos, melhorando a comunicação entre as equipes através de artefatos com informações relevantes tanto para desenvolvimento quanto para teste, oferecendo visibilidade sobre as disciplinas em execução, os envolvidos e suas responsabilidades.

A partir da análise dos processos realizada no Capítulo 3, pode-se observar que mesmo os processos que apresentam artefatos, responsabilidades e atividades bem definidas, possibilitando adequação às necessidades específicas, eles não contemplam as peculiaridades da

estratégia de desenvolvimento distribuídas. Além disso, com base na análise dos trabalhos relacionados pode-se extrair requisitos para a definição de um processo compatível com as características do DDS.

Com base na lacuna identificada foi elaborada uma abordagem integrada de desenvolvimento e teste de software. A abordagem abrange um conjunto de atividades que devem ocorrer ao longo do ciclo de vida de um projeto e as características essenciais de cada uma. Considera atividades desde o modelo de negócios a ser adotado para atuar em DDS (empresas globais, parcerias estratégicas, entre outros) até a implementação. A modelagem da abordagem foi realizada utilizando a notação SPEM com o intuito de facilitar a comunicação, o entendimento do processo, o reuso, apoiar a sua evolução, facilitar o gerenciamento e com vistas a melhoria contínua do processo.

7.1 Contribuições

Como contribuições desse trabalho pode-se destacar:

- a abordagem encontra-se estruturada em termos de papéis, artefatos e atividades bem definidos, o que auxilia diretamente na sincronização, por oferecer a todos os envolvidos uma nomenclatura comum. Isso possibilita uma melhor compreensão dos termos do domínio de negócios e os marcos do projeto apesar de suas diferenças em termos de cultura e estrutura organizacional, que podem ocorrer em um cenário de DDS. Além disso, ter um processo bem definido permite que o mesmo seja analisado e como resultado dessa análise, ele possa sofrer evoluções, sendo passível de melhoria contínua;

- oferece suporte ao processo de desenvolvimento, planejamento e projeto de software à medida que apresenta atividades e diretrizes tanto no nível estratégico (aspectos de gerência) como operacional (verificação e validação), os quais fornecem subsídios para a qualidade do software a ser desenvolvido;

- utiliza o gerenciamento híbrido (centralizado-descentralizado) em relação ao controle das atividades e enfatiza a importância da definição do Gerente de Projeto Local, bem como define suas atividades, para estimular a confiança e compromisso entre os membros;

- apresenta atividades e diretrizes para a definição do idioma a ser adotado na formalização do processo e comunicação entre as equipes;

- formaliza a importância do teste de software dentro de um projeto, tratando-o como uma abordagem que deve ser instanciada em paralelo ao desenvolvimento e com atividades desde a concepção do software.

- integração das abordagens de desenvolvimento e teste o que confere facilidade para realizar alterações nos requisitos devido à rastreabilidade provida.

- oferece mecanismos para a padronização e redução da imprecisão dos artefatos. Com a padronização dos artefatos é possível reduzir a comunicação entre as equipes e a especificação de *templates* é um fator primordial para facilitar a efetiva comunicação entre os membros das equipes.

- utiliza a notação UML, que é de fácil entendimento tanto para o meio acadêmico quanto industrial e possue semântica para transmitir informações aos desenvolvedores;

- utiliza especificação formal de testes através do perfil U2TP para amenizar os problemas de ambiguidade e reduzir a necessidade de comunicação;

- oferece diretrizes para nortear o gerente de projetos (Local e Global) na distribuição das atividades;

- especifica atividades que tratam da definição de infraestrutura para possibilitar a colaboração, controle da documentação e controle de versão dos artefatos;

- auxilia na percepção e conhecimento das atividades desenvolvidas, bem como o que será realizado em cada local e as responsabilidades de cada membro, por meio de duas categorias de percepção:

  - percepção de atividade: procura responder questões, tais como: "Quem está trabalhando em qual atividade?", "Quem foi o responsável por determinada tarefa e qual o resultado? "Onde a atividade está sendo executada?"
  - percepção de processo: relaciona-se a questões do tipo: "Como a tarefa daquele pessoa se encaixa na minha? "O que devo fazer agora?"

7.2 Dificuldades e Limitações

Durante a realização desta dissertação foram encontradas algumas dificuldades e limitações, sendo elas:

- no âmbito regional, há dificuldade em encontrar empresas que utilizem essa configuração de desenvolvimento.
- os projetos desenvolvidos utilizando a estratégia distribuída, geralmente, são de grande porte, o que demandaria um prazo maior para o acompanhamento do projeto.
- dificuldade de reunir equipes dispersas, mesmo no meio acadêmico, que tenham disponibilidade para participar do experimento.

7.3 Trabalhos Futuros

Com base nas limitações apresentadas e para dar continuidade as atividades realizadas neste trabalho, foram identificadas algumas sugestões de trabalhos futuros, os quais estão além do escopo desta dissertação:

- replicar o estudo de viabilidade para ampliar a aquisição de conhecimento sobre a aplicação do processo, a credibilidade e o índice de confiabilidade do mesmo.
- refinar a abordagem proposta por meio da condução dos seguintes estudos:
    1. estudo de observação: para adquirir uma compreensão refinada sobre a tecnologia e averigurar se a aplicação prática da abordagem faz sentido.
    2. estudo de caso (ciclo de vida): para caracterizar a aplicação da abordagem no contexto de um ciclo de vida de desenvolvimento e avaliar o efeito de sua aplicação com o intuito de verificar se o abordagem é adequada no contexto de um ciclo de vida.
    3. estudo de caso (indústria): para determinar a viabilidade prática da aplicação do abordagem, aprimorar o entendimento dos pesquisadores resultando no refinamento da abordagem com o objetivo de analisar se a abordagem é adequada no contexto industrial
- definir mecanismos para analisar as redes sociais formadas durante o projeto;
- definir métricas para avaliar a produtividade, a comunicação e a qualidade dos artefatos gerados;
- integrar a abordagem apresentada ao DiSEN *Distributed Software Engineering Environment*, que é um ADDS (Ambiente de Desenvolvimento Distribuído de Software), que visa disponibilizar ferramentas de apoio à comunicação, à persistência e à

cooperação para equipes de desenvolvimento geograficamente dispersas (Huzita et al., 2007).

7.4 Publicações

No decorrer do mestrado foram publicados os seguintes artigos:

- LEAL, G. C. L. ; SILVA, C. A ; HUZITA, H. M. M. ; TAIT, T. F. C. . *Towards a Process to Information System Development with Distributed Teams. In: 12th International Conference on Enterprise Information Systems*, 2010, Funchal.

- LEAL, G. C. L. ; SILVA, C. A ; HUZITA, H. M. M. ; TAIT, T. F. C. . Uma Análise Sociotécnica de Processos de Software sob a Perspectiva do Desenvolvimento com Equipes Distribuídas. In: VI Workshop, 2010, Belém. VI Workshop, 2010.

- LEAL, G. C. L. ; SILVA, C. A ; HUZITA, H. M. M.. Um Processo de Desenvolvimento de Produto de Software para a Estratégia de Produção Distribuída. In: 5th *Americas International Conference on Production Research (ICPR Americas)*, Bogotá, Colômbia, 2010.

- LEAL, G. C. L. ; SILVA, C. A ; DELAMARO, M. E. ; HUZITA, H. M. M. . *A Proposal an Integrated Approach of Software Development and Testing to Distributed Teams. In: International Conference IADIS Information Systems* 2010, 2010, Porto.

- LEAL, G. C. L. ; SILVA, C. A ; HUZITA, H. M. M. . *Towards a Distributed Software Process. In: IADIS International Conference IADIS Information Systems* 2010, 2010, Porto.

Referências Bibliográficas

Abdurazik, A. Evaluation of three specification-based testing criteria. In: *ICECCS '00: Proceedings of the 6th IEEE International Conference on Complex Computer Systems*, Washington, DC, USA: IEEE Computer Society, 2000, p. 179.

Al-Asmari, K.; Yu, L. Experiences in distributed software development with wiki. In: *Software Engineering Research and Practice*, 2006, p. 389–293.

Alves, E. L. G.; Machado, P. D. L.; Ramalho, F. Uma abordagem integrada para desenvolvimento e teste dirigido por modelos. In: *2nd Brazilian Workshop on Systematic and Automated Software Testing*, Campinas, SP, 2008, p. 74–83.

Audy, J.; Prikladnicki, R. *Desenvolvimento distribuído de software: Desenvolvimento de software com equipes distribuídas.* Rio de Janeiro, RJ: Elsevier, 2008.

Avritzer, A.; Hasling, W.; Paulish, D. Process investigations for the global studio project version 3.0. In: *ICGSE '07: Proceedings of the International Conference on Global Software Engineering*, Washington, DC, USA: IEEE Computer Society, 2007, p. 247–251.

Avritzer, A.; Paulish, D.; Cai, Y. Coordination implications of software architecture in a global software development project. In: *WICSA '08: Proceedings of the Seventh Working IEEE/IFIP Conference on Software Architecture (WICSA 2008)*, Washington, DC, USA: IEEE Computer Society, 2008, p. 107–116.

Basili, V. R. The role of controlled experiments in software engineering research. In: *Empirical Software Engineering Issues*, 2006, p. 33–37.

Bazman The benefits of outsourced testing. http://bazman.tripod.com/outsourced.html, 2007.

Ben-Shaul, I. Z.; Kaiser, G. E. A paradigm for decentralized process modeling and its realization in the oz environment. In: *ICSE '94: Proceedings of the 16th*

international conference on Software engineering, Los Alamitos, CA, USA: IEEE Computer Society Press, 1994, p. 179–188.

Berger, P. M. *Instanciação de processos de software em ambientes configurados na estação taba.* Dissertação de Mestrado, COOPE, Universidade Federal do Rio de Janeiro, Rio de Janeiro - Rio de Janeiro, 2003.

Biolchini, J.; Mian, P.; Natali, A.; Travassos, G. *Systematic review in software engineering: Relevance and utility.* Relatório Técnico, COPPE/UFRJ, Rio de Janeiro, RJ, 2005.

Briand, L. C.; Labiche, Y. A uml-based approach to system testing. In: *Proceedings of the 4th International Conference on The Unified Modeling Language, Modeling Languages, Concepts, and Tools*, London, UK: Springer-Verlag, 2001, p. 194–208.

Carmel, E. *Global software teams: collaborating across borders and time zones.* Upper Saddle River, NJ, USA: Prentice Hall PTR, 1999.

Carmel, E.; Tija, P. *Offshoring information technology: Sourcing and outsourcing to a global workforce.* New York: Cambridge University Press, 2005.

Carniello, A. *Teste baseado na estrutura de casos de uso.* Dissertação de Mestrado, Universidade Estadual de Campinas, Campinas, SP, 2003.

Cartaxo, E. G. *Geração de casos de teste funcional para aplicações de celulares.* Dissertação de Mestrado, Universidade Federal de Campina Grande (UFCG), Campina Grande - Paraíba, 2006.

Cibotto, R. A. G.; Pagno, R. T.; Tait, T. F. C.; Huzita, E. H. M. Uma análise da dimensão sócio-cultural no desenvolvimento distribuído de software. In: *V Workshop Um Olhar Sociotécnico sobre a Engenharia de Software WOSES*, Ouro Preto, Minas Gerais, 2009.

Crespo, A. N.; Jino, M. *Processo de testde de software.* Relatório Técnico, Centro de Tecnologia da Informação Renato Archer (CenPRA), 2005.

Crespo, A. N.; Silva, O. J.; Borges, C. A.; Salviano, C. F.; Argollo, M. T.; Jino, M. Uma metodologia de teste de software no contexto de melhoria de processo. In: *III Simpósio Brasileiro de Qualidade de Software SBQS*, Brasília-DF, 2004.

Dalal, S. R.; Jain, A.; Karunanithi, N.; Leaton, J. M.; Lott, C. M.; Patton, G. C.; Horowitz, B. M. Model-based testing in practice. In: *ICSE '99: Proceedings of the 21st international conference on Software engineering*, New York, NY, USA: ACM, 1999, p. 285–294.

Damian, D. Workshop on global software development. In: *ICSE '02: Proceedings of the 24th International Conference on Software Engineering*, New York, NY, USA: ACM, 2002, p. 667–668.

Damian, D.; Lanubile, F. The 3rd international workshop on global software development. In: *ICSE '04: Proceedings of the 26th International Conference on Software Engineering*, Washington, DC, USA: IEEE Computer Society, 2004, p. 756–757.

Delamaro, M. E.; Maldonado, J. C.; Jino, M. *Introdução ao teste de software.* 1 ed. Editora Campus, 2007.

El-Far, I. K.; Whittaker, J. A. Model-based software testing. 2001.

Enami, L. N. M. *Um modelo de gerenciamento de projetos para um ambiente de desenvolvimento distribuído de software.* Dissertação de Mestrado, Universidade Estadual de Maringá (UEM), Maringá - Paraná, 2006.

Fantinato, M. *Geração de casos de teste funcional para aplicações de celulares.* Dissertação de Mestrado, Faculdade de Engenharia Elétrica e de Computação - Universidade Estadual de Campinas (FEEC/UNICAMP), Campinas - São Paulo, 2002.

Fuggetta, A. Software process: a roadmap. In: *ICSE '00: Proceedings of the Conference on The Future of Software Engineering*, New York, NY, USA: ACM, 2000, p. 25–34.

Guimarães, D. C. *Ster: Uma estratégia de testes para sistemas reativos.* Dissertação de Mestrado, Universidade Estadual de Campinas (Unicamp), Campinas - São Paulo, 2005.

Hargreaves, E.; Damian, D. Can global software teams learn from military teamwork models? In: *Proc. of 3rd Int. Workshop on Global Software Development, ICSE'04*, 2004.

HARTMANNM, J.; VIEIRA, M.; RUDER, A. Uml-based test generation and execution. In: *Proceedings of the 21st Workshop on Software Test, Analyses and Verification (GI-FG TAV)*, Berlin, 2004.

HERBSLEB, J. D.; MOCKUS, A.; FINHOLT, T. A.; GRINTER, R. E. Distance, dependencies, and delay in a global collaboration. In: *CSCW '00: Proceedings of the 2000 ACM conference on Computer supported cooperative work*, New York, NY, USA: ACM, 2000, p. 319–328.

HUMPHREY, W. S.; KELLNER, M. I. Software process modeling: principles of entity process models. In: *ICSE '89: Proceedings of the 11th international conference on Software engineering*, New York, NY, USA: ACM, 1989, p. 331–342.

HUZITA, E. H. M.; SILVA, C. A.; WIESE, I. S.; TAIT, T. F. C.; QUINAIA, M.; SCHIAVONE, F. L. Um conjunto de soluções para apoiar o desenvolvimento distribuído de software. In: *II Workshop de Desenvolvimento Distribuído de Software*, Campinas, SP, 2008, p. 101–110.

HUZITA, E. H. M.; TAIT, T. F. C.; COLANZI, T. E.; QUINAIA, M. Um ambiente de desenvolvimento distribuído de software - disen. In: *I Workshop de Desenvolvimento Distribuído de Software*, João Pessoa, PB, 2007, p. 31–38.

IEEE Ieee standard for software test documentation. ANSI/IEEE Std 829-1998, 1998.

JACOBSON, I. *Object oriented software engineering.* Addison-Wesley, 1992.

JIMÉNEZ, M.; PIATTINI, M.; VIZCAÍNO, A. Challenges and improvements in distributed software development: A systematic review. *Advances in Software Engineering*, v. vol. 2009, 2009.

JURISTO, N.; MORENO, A. M.; VEGAS, S. Reviewing 25 years of testing technique experiments. *Empirical Softw. Engg.*, v. 9, n. 1-2, p. 7–44, 2004.

KITCHENHAM, B. A. *Procedures for undertaking systematic reviews.* Relatório Técnico, Computer Science Department, Keele University, 2004.

KITCHENHAM, B. A.; PFLEEGER, S. L.; PICKARD, L. M.; JONES, P. W.; HOAGLIN, D. C.; EMAM, K. E.; ROSENBERG, J. Preliminary guidelines for empirical research in software engineering. *IEEE Trans. Softw. Eng.*, v. 28, n. 8, p. 721–734, 2002.

KRUCHTEN, P. *The rational unified process: An introduction.* Addison-Wesley, 2003.

Layman, L.; Williams, L.; Damian, D.; Bures, H. Essential communication practices for extreme programming in a global software development team. *Information and Software Technology*, v. 48, n. 9, p. 781–794, 2006.
Disponível em `http://www.sciencedirect.com/science/article/B6V0B-4JF8H93-2/2/625aa9c8d29f7c60b2a5a38425f9a41f`

Leal, G. C. L.; Silva, C. A.; Huzita, E. H. M. Towards a distributed software process. In: *International Conference IADIS Information Systems 2010*, Porto, Portugal, 2010.

L'Erário, A. As alianças estratégicas no desenvolvimento distribuído de software. In: *XXIX Encontro Nacional de Engenharia de Produção - A Engenharia de Produção e o Desenvolvimento Sustentável: Integrando Tecnologia e Gestão*, Salvador, BA, 2009a, p. 756–757.

L'Erário, A. *M3ds: um modelo de dinâmica de desenvolvimento distribuído do software.* Tese de Doutoramento, Escola Politécnica da Universidade de São Paulo, São Paulo, SP, 2009b.

Lings, B.; Lundell, B.; Agerfalk, P. J.; Fitzgerald, B. A reference model for successful distributed development of software systems. In: *ICGSE '07: Proceedings of the International Conference on Global Software Engineering*, Washington, DC, USA: IEEE Computer Society, 2007, p. 130–139.

Mafra, S. N.; Barcelos, R. F. T. G. H. Aplicando uma metodologia baseada em evidÊncias na definição de novas tecnologias de software. In: *XX Simpósio Brasileiro de Engenharia de Software (SBES)*, Florianopólis, SC, 2006, p. 239–254.

Mantovan, U. *Uma abordagem, baseada em framework e na tÉcnica de descriÇÃo formal estelle, para o desenvolvimento de sistemas de arquivos paralelos distribuÍdos.* Dissertação de Mestrado, Escola Politécnica, Universidade de São Paulo (POLI/USP), São Paulo - São Paulo, 2006.

Mats, L. The top five software-testing problems and how to avoid them. Http://pe2bz.philpem.me.uk/Misc/-Testing.pdf, 2001.

McGregor, J. D.; Sykes, D. A. *A practical guide to testing object-oriented software.* Addison-Wesley, 2001.

Mockus, A.; Herbsleb, J. Challenges of global software development. In: *METRICS '01: Proceedings of the 7th International Symposium on Software Metrics*, Washington, DC, USA: IEEE Computer Society, 2001, p. 182.

Montgomery, D. C.; Runger, G. C. *Estatística aplicada e probabilidade para engenheiros*. 4 ed. Norwell, MA, USA: LTC, 2009.

Mullick, N.; Bass, M.; Houda, Z.; Paulish, P.; Cataldo, M. Siemens global studio project: Experiences adopting an integrated gsd infrastructure. In: *Global Software Engineering, 2006. ICGSE '06. International Conference on*, 2006, p. 203–212.

Myers, G. *The art of software testing*. John Willey, 2001.

Offutt, A. J.; Liu, S.; Abdurazik, A.; Ammann, P. Generating test data from state-based specifications. *Softw. Test., Verif. Reliab.*, v. 13, n. 1, p. 25–53, 2003. Disponível em `http://dblp.uni-trier.de/db/journals/stvr/stvr13.html#OffuttLAA03`

OMG *Uml 2.0 testing profile specification*. Relatório Técnico, 2004.

OMG Software process engineering metamodel specification. An Adopted Specification of the Object Management Group, Inc.T, 2005.

Paasivaara, M.; Lassenius, C. Using iterative and incremental processes in global software development. *IEEE Seminar Digests*, v. 2004, n. 912, p. 42–47, 2004. Disponível em `http://link.aip.org/link/abstract/IEESEM/v2004/i912/p42/s1`

Packevicius, S.; Usaniov, A. B. E. Software testing using imprecise ocl constraints as oracles. In: *Proceedings of the 2007 international Conference on Computer Systems and Technologies (CompSystech'07)*, New York, NY, USA: ACM, 2007, p. 1–6.

Paulish, D. J. Scalable distributed organizations for ultra-large-scale software. In: *ULS '07: Proceedings of the International Workshop on Software Technologies for Ultra-Large-Scale Systems*, Washington, DC, USA: IEEE Computer Society, 2007, p. 4.

Ribeiro, M. B.; Czekster, R. M.; Webber, T. Improving productivity of local software development teams in a global software development environment. In: *Global Software Engineering, 2006. ICGSE '06. International Conference on*, 2006, p. 253–254.

Rocha, R.; Arcoverde, D.; Brito, D.; Arôxa, B.; Costa, C.; Silva, F. Q. B.; J., A.; Meira, S. R. L. Uma experiência na adaptação do rup em pequenas equipes

de desenvolvimento distribuído. In: *II Workshop de Desenvolvimento Distribuído de Software*, Campinas, SP, 2008, p. 81–90.

Rorato, T. *Em direção a uma semântica da linguagem de descrição de reuso em uml/ocl.* Dissertação de Mestrado, Pontifícia Universidade Católica do Rio Grande do Sul, Porto Alegre - Rio Grande do Sul, 2007.

Sangwan, R.; Bass, M.; Mullick, N.; Paulish, D. J.; Kazmeier, J. *Global software development handbook (auerbach series on applied software engineering series).* Boston, MA, USA: Auerbach Publications, 2006.

Sengupta, B.; Chandra, S.; Sinha, V. A research agenda for distributed software development. In: *ICSE '06: Proceedings of the 28th international conference on Software engineering*, New York, NY, USA: ACM, 2006, p. 731–740.

Shull, F.; Carver, J.; Travassos, G. H. An empirical methodology for introducing software processes. *SIGSOFT Softw. Eng. Notes*, v. 26, n. 5, p. 288–296, 2001.

Silva, D.; Lopes, M. C. M.; Lopes, G. O. Sitest: Uma metodologia de teste de software com base no modelo cmmi. In: *III Simpósio Brasileiro de Sistemas de Informação*, Curitiba, PR, 2006, p. 11–11.

Souza, E. P. R.; Gusmão, C. M. G. Rbtprocess - modelo de processo de teste de software baseado em riscos. In: *13 Workshop de Teses e Dissertações em Engenharia de Software*, Campinas, São Paulo, 2008.

Taylor, P. S.; Greer, D.; Sage, P.; Coleman, G.; McDaid, K.; Keenan, F. Do agile gsd experience reports help the practitioner? In: *GSD '06: Proceedings of the 2006 international workshop on Global software development for the practitioner*, New York, NY, USA: ACM, 2006, p. 87–93.

Travassos, G. H. *Introdução à engenharia de software experimental.* Relatório Técnico ES-590/02, Programa de Engenharia de Sistemas e Computação, COPPE/UFRJ, 2002.

Urdangarim, R. *Uma investigação sobre o uso de práticas extreme programming no desenvolvimento global de software.* Dissertação de Mestrado, Pontifícia Universidade Católica do Rio Grande do Sul, Porto Alegre - Rio Grande do Sul, 2008.

VANZIN, M.; RIBEIRO, M.; PRIKLADNICKI, R.; CECCATO, I.; ANTUNES, D. Global software processes definition in a distributed environment. In: *Software Engineering Workshop, 2005. 29th Annual IEEE/NASA*, IEEE Computer Society, 2005, p. 57–65.

WEISSLEDER, S.; SCHLINGLOFF, B.-H. Quality of automatically generated test cases based on ocl expressions. In: *ICST '08: Proceedings of the 2008 International Conference on Software Testing, Verification, and Validation*, Washington, DC, USA: IEEE Computer Society, 2008, p. 517–520.

WOHLIN, C.; RUNESON, P.; HÖST, M.; OHLSSON, M. C.; REGNELL, B.; WESSLÉN, A. *Experimentation in software engineering: an introduction*. Norwell, MA, USA: Kluwer Academic Publishers, 2000.

Apêndice

A

Protocolo da Revisão Sistemática

A.1 Introdução

Uma revisão sistemática atua como um meio para identificar, avaliar e interpretar toda pesquisa disponível e relevante sobre uma questão de pesquisa, um tópico ou um fenômeno de interesse. A condução de uma revisão sistemática supostamente apresenta uma avaliação justa do tópico de pesquisa, na medida que utiliza uma metodologia de revisão rigorosa, confiável e passível de auditagem (Kitchenham, 2004).

As principais razões para se realizar uma revisão sistemática são: (i) resumir a evidência existente a respeito de uma determinada tecnologia; (ii) identificar lacunas na pesquisa atual a fim de sugerir áreas para investigações futuras; e, (iii) fornecer um *framework* para direcionar atividades novas de pesquisa.

Este documento apresenta o protocolo de revisão sistemática que foi conduzido para identificar os estudos que abordam processos de software utilizados no desenvolvimento com equipes distribuídas e as necessidades dessa abordagem no que se refere ao processo de desenvolvimento.

A.2 Formulação da Pergunta

A questão primária formulada no planejamento da revisão sistemática foi "Quais são os processos de software utilizados no desenvolvimento distribuído?". Como questão

secundária tem-se: "Quais as peculiaridades do desenvolvimento distribuído que devem ser contempladas pelos processos de desenvolvimento?".

Como critérios de inclusão que atendessem à questão primária foram considerados os trabalhos que fizessem referência a processos ou adaptações de processos de software utilizados com equipes distribuídas. Para atender à questão secundária foram considerados os trabalhos que retratassem a necessidade dos processos de desenvolvimento contemplarem as peculiaridades do desenvolvimento distribuído.

Os critérios de exclusão utilizados para a questão primária foram: trabalhos que não abordassem de processos de desenvolvimento de software; não estivessem redigidos em inglês; não correspondessem a trabalho de pesquisa; ou que não era possível ter acesso a versão completa do trabalho. Para responder a questão secundária, não foram considerados os trabalhos que não retratavam as características da abordagem distribuída de desenvolvimento.

A.3 Seleção das Fontes de Pesquisa

As fontes consideradas para essa revisão foram bases de dados eletrônicas indexadas (IEEE, ACM e Springer). Dessas fontes, foram considerados apenas os trabalhos redigidos em inglês. As palavras-chaves utilizadas foram: *"software process"*, *"software development"*, *"virtual teams"*, *"distributed teams"*, *"global"* e *"distributed"*. Por meio da combinação desses termos foram elaboradas as *strings* de busca. Como as máquinas de busca apresentam particularidades relacionadas à construção de *strings* de busca, quando necessário, foram realizadas adaptações.

As buscas foram feitas em bases de dados eletrônicas indexadas por meio de *strings* de busca. A execução da busca foi realizada de forma incremental, sendo realizado um incremento para cada fonte de busca. Nesta etapa foram recuperados um total de 49 trabalhos.

A.4 Seleção Preliminar

Nessa etapa foi realizada a leitura dos resumos dos trabalhos e, quando necessário, algumas seções do artigo como introdução ou conclusão. Não foram estabelecidos critérios específicos de qualidade para determinar a validade dos artigos selecionados. Nesse estudo inicial, o fato dos trabalhos serem procedentes de fontes reconhecidas bastariam para aceitação da validade.

A.5 Seleção Final e Extração dos Dados

Nessa etapa foi realizada a leitura completa dos trabalhos e os selecionados foram sintetizados, extraindo dados para posterior análise e interpretação dos estudos obtidos. Segundo Biolchini et al. (2005), o foco principal dessa fase é sintetizar os estudos válidos de modo que, posteriormente, possam ser realizadas generalizações. Os dados extraídos foram:

- citação do trabalho selecionado;
- fonte de busca em que o trabalho foi recuperado;
- processo identificado, em resposta a questão primária da revisão sistemática;
- elementos descritos, apresenta os elementos componentes de um processo e as peculiaridades da abordagem de desenvolvimento distribuído que foram identificadas no trabalho selecionado. Essas particularidades são referentes à definição de:
    1. mecanismos para explicitar as diferenças (cultural, fuso-horário e idioma) entre as equipes envolvidas no projeto;
    2. especificação formal dos requisitos para amenizar os problemas de ambiguidade e imprecisão dos artefatos;
    3. estratégias para realizar a distribuição do desenvolvimento;
    4. critérios para suavizar a integração dos módulos desenvolvidos em locais separados, a fim de reduzir os erros de integração.

A.6 Considerações Finais

Este documento apresentou o protocolo da revisão sistemática conduzida para verificar os processos de desenvolvimento de software que vem sendo utilizados com equipes distribuídas. Os estudos selecionados foram, na sua maior parte, referentes aos problemas (diversidade cultural, dispersões geográficas, coordenação e comunicação) encontrados quando da adoção de desenvolvimento distribuído.

Os dados obtidos, com a revisão sistemática, forneceram subsídios para a definição da abordagem proposta nesta dissertação. Um detalhamento maior sobre a revisão sistemática pode ser visto em Leal et al. (2010).

Apêndice

B

Templates Abordagem de Desenvolvimento

Esse documento apresenta *templates* para os artefatos gerados pela abordagem de desenvolvimento proposta.

B.1 Plano de Desenvolvimento de Software

Esta seção apresenta um template para os artefatos Plano de Desenvolvimento Global e Plano de Desenvolvimento Local, conforme pode ser visto nas Tabelas B.1 e B.2. Estes artefatos são gerados na Disciplina Planejamento.

B.2 Descrição Extendida Casos de Uso

A Tabela B.3 ilustra o *template* para a descrição de casos de uso, artefato gerado na Disciplina Requisitos.

Tabela B.1: Plano de Desenvolvimento de Software Global

Identificação do Projeto:	
Finalidade:	
Escopo:	
Restrições:	
Site Central:	
Idioma:	
Fuso-horário:	
Gerente de Projetos Global:	
Idioma do Projeto:	
Configuração do Desenvolvimento:	
Planejamento	
Requisitos	
Desenvolvimento	
Implementação	
Estratégia de distribuição (disciplina, componente, caso de uso, método):	

Sites envolvidos

Site	Gerente de Projeto Local	Diferença fuso-horário	Idioma

Estrutura organizacional:	

Papéis e responsabilidades:

Site	Disciplina	Função

Recursos (Hardware e Software):	
Ferramentas de Comunicação:	
Cronograma:	

Tabela B.2: Plano de Desenvolvimento de Software Local

Gerente de Projetos Local:	
Escopo:	
Restrições:	
Idioma do Projeto:	
Disciplina:	
Estrutura organizacional :	
Papéis e responsabilidades:	

Membro	Papel	Atividade

Recursos (Hardware e Software):	
Ferramentas de Comunicação:	
Cronograma:	

Tabela B.3: Descrição de Caso de Uso

Nome do Caso de Uso	
Autor/Equipe	
Data	
Atores	
Descrição	
Pré-condições	
Pós-condições	
Fluxo de eventos	
Fluxos Alternativos	

Apêndice

C

Templates Abordagem de Teste

Esse documento apresenta *templates* para os artefatos gerados pela abordagem de teste proposta.

C.1 Plano de Testes

O *template* para o Plano de Testes é apresentado na Tabela C.1.

Tabela C.1: Plano de Testes

Identificação do Projeto:

Finalidade:

Escopo:

Funcionalidades:

Configuração do Teste:

Planejamento	
Preparação	
Projeto	
Execução	

Estratégia de distribuição (disciplina, casos de teste):

Sites envolvidos

Site	Gerente de Projeto Local	Diferença fuso-horário	Idioma

Papéis e responsabilidades:

Site	Disciplina	Função

Recursos (Hardware e Software):	
Cronograma:	

C.2 Especificação de Projeto de Teste

Na Tabela C.2 é ilustrado o *template* para a Especificação de Projeto de Teste.

Tabela C.2: Especificação de Projeto de Teste

Identificação do Projeto:			
Responsável:			
Site	**Gerente de Projeto Local**	**Diferença fuso-horário**	**Idioma**
Técnicas de teste:			
Critérios de teste:			
Critérios de seleção de casos de teste:			
Critérios de parada:			
Ferramentas de Teste:			

C.3 Especificação de Casos de Teste

O *template* para a Especificação de Casos de Teste, artefato produzido na Disciplina de Projeto, é mostrado na Tabela C.3

Tabela C.3: Especificação de Casos de Teste

Identificação do Projeto:			
Responsável:			
Site	**Gerente de Projeto Local**	**Diferença fuso-horário**	**Idioma**
Requisitos de ambiente:			
CT	**Pacote**	**Descrição**	**Dependência**

C.4 Especificação de Procedimentos de Teste

A Tabela C.4 mostra o *template* para o artefato Especificação de Procedimentos de Teste.

Tabela C.4: Especificação de Procecimentos de Teste

Identificação do Projeto:			
Responsável:			
Site	**Gerente de Projeto Local**	**Diferença fuso-horário**	**Idioma**
Requisitos de ambiente:			
CT	**Roteiro de Teste (passos)**	**Resultado Esperado**	

C.5 Diário de Testes

O *template* para o Diário de Testes é apresentado na Tabela C.5.

Tabela C.5: Diário de Testes

Identificação do Projeto:				
Responsável:				
Site	**Gerente de Projeto Local**	**Diferença fuso-horário**	**Idioma**	
Ambiente de execução (hardware):				
CT	**Responsável pela execução**	**Resultado**	**Eventos não esperados**	**Data**

C.6 Relatório de Incidentes

O *template* para o Relatório de Incidentes, artefato produzido na Disciplina de Execução, é mostrado na Tabela C.6

Tabela C.6: Relatório de Incidentes

Identificação do Projeto:					
Responsável:					
Site	**Gerente de Projeto Local**		**Diferença fuso-horário**		**Idioma**
Status	**CT**	**Prioridade**	**Responsável pela correção**	**Descrição do Erro**	**Data da Correção**

C.7 Resumo de Testes

A Tabela C.7 ilustra o *template* do artefato Resumo de Testes.

Tabela C.7: Resumo de Testes

Identificação do Projeto:			
Responsável:			
Site	**Gerente de Projeto Local**	**Diferença fuso-horário**	**Idioma**
Início do Testes:		**Fim dos Testes:**	
Descrição Detalhada dos Testes			
Números do Teste			
Casos de Teste criados antes do início dos testes			
Casos de Teste criados durante os testes			
Casos de Teste executados			
Casos de Teste com sucesso			
Casos de Teste com erros			
Casos de Teste enviados para correção			
Percentuais do Teste			
Casos de Teste Executados			
Casos de Teste Executados com Sucesso			
Casos de Teste Executados com Incidência de Erros			
Casos de Teste Corrigidos			

Apêndice

D

Artefatos

D.1 Introdução

Este documento apresenta artefatos gerados durante a instanciação da abordagem no cenário proposto no Capítulo 5, o qual é utilizado como prova de conceito para a abordagem.

D.2 Descrição Textual

O MSPaper objetiva gerenciar o processo de submissão e avaliação de artigos em eventos científicos, sob a perspectiva de quatro tipos de usuários: (i) presidente do comitê científico; (ii) revisores; (iii) autores; e, (iv) administrador. As funcionalidades a serem desenvolvidas são:

- Registro do evento - consiste em caracterizar o evento através da definição de questões relacionadas à: temas que os artigos submetidos devem atender, número de revisões por artigo, calendário do processo de seleção, incluindo: prazo de submissão, prazo de devolução das revisões pelos revisores, data de notificação da aceitação ou rejeição, data limite para submissão da versão final dos artigos aceitos e percentual de artigos a serem aceitos.

- Submissão de artigos - um dos autores submete o arquivo do artigo em formato PDF. Nessa etapa são identificados todos os autores, a área do artigo, as palavras chaves e o resumo.

- Distribuição dos artigos - o presidente do comitê científico é responsável por constituir o comitê de programa com base nas especialidades de conhecimento. Após isso, deve liberar a distribuição dos artigos para os autores da área relacionada.

- Revisão dos artigos - os revisores analisam os artigos submetidos segundo os critérios de avaliação, que são: relevância, conteúdo técnico e rigor científico, originalidade, qualidade da escrita e avaliação global.

- Consolidação dos resultados - o presidente do comitê científico acompanha as revisões, verificando se todos as revisões foram devolvidas e se houveram atrasos. Além disso, resolve os possíveis conflitos de avaliação, define o número de artigos aceitos e encaminha a notificação de aceitação ou rejeição aos autores.

- Controle de Acesso - consiste em definir os níveis de acesso de acordo com o tipo de usuário.

O administrador é responsável pelas configurações relacionadas ao controle de acesso dos usuários. O presidente do comitê científico é o responsável pelo registro do evento no MSPaper, distribuição dos artigos entre os revisores e a consolidação dos resultados.

Os autores podem realizar a submissão de artigos e os revisores são responsáveis pela avaliação dos artigos que lhes forem atribuídos.

D.3 Especificação dos Casos de Uso

Esta seção apresenta, para cada caso de uso do MSPaper, sua especificação, contendo descrição, fluxo de eventos, pré e pós-condições.

Tabela D.1: Gerenciar acesso.

Nome do Caso de Uso	Gerenciar acesso
Autor/Equipe	Gislaine Camila/Maringá
Data	12/03/2010
Atores	Administrador e PresidenteComitêCientífico
Descrição	Parte do sistema responsável por gerenciar os níveis de acesso dos usuários.
Pré-condições	Usuário cadastrado no sistema.
Pós-condições	O usuário logado no sistema ou não.
Fluxo de eventos	1. O usuário informa os dados de login e senha. 2. A tela principal do MSPaper é aberta.
Tratamento de exceções	1. Login e/ou senha inválidos. (a) O sistema emite uma mensagem informando que os dados estão incorretos.

Tabela D.2: Gerenciar evento.

Nome do Caso de Uso	Gerenciar evento
Autor/Equipe	Gislaine Camila/Maringá
Data	12/03/2010
Atores	Administrador e PresidenteComitêCientífico
Descrição	O ator Administrador cadastra o evento e vincula o usuário PresidenteComitêCientífico ao evento.
Pré-condições	Usuário estar logado no sistema e ter permissão de Administrador ou ter permissão de PresidenteComitêCientífico.
Pós-condições	Evento cadastrado.
Fluxo de eventos	1. O Administrador cadastra o evento e informa o usuário que será o Presidente do Comitê Científico. 2. O usuário PresidenteComitêCientífico insere as demais informações e disponibiliza o evento para que os demais usuários possam visualizar.
Tratamento de exceções	1. Campo número de revisões por artigo está vazio ou com valor menor ou igual a zero. (a) O sistema emite uma mensagem solicitando que seja informado um valor maior que zero. 2. O prazo de submissão é menor que a data atual. (a) O sistema emite uma mensagem informando que o prazo de submissão deve ser posterior data atual. 3. A data de devolução das submissões é menor que a data de submissão. (a) O sistema emite uma mensagem informando que a data de devolução das revisões deve ser maior que a data de submissão. 4. A data de notificação é menor que a data atual ou menor que a data de devolução das revisões. (a) O sistema emite uma mensagem informando que a data de notificação deve ser maior que a data de devolução das revisões. 5. O percentual de artigos aceitos está vazio ou é menor ou igual a zero. (a) O sistema emite uma mensagem solicitando que seja informado um valor maior que zero.

Tabela D.3: Submeter artigo.

Nome do Caso de Uso	Submeter artigo
Autor/Equipe	Gislaine Camila/Maringá
Data	12/03/2010
Atores	Autor
Descrição	O autor realizará a submissão de um artigo, para isso ele deve informar a qual tema do evento o artigo está relacionado, os autores, título, resumo, palavras-chave e fazer o *upload* do arquivo com extensão PDF.
Pré-condições	O usuário deve estar cadastrado no MSPaper e o prazo para submissão de artigos deve estar aberto.
Pós-condições	A confirmação de submissão é encaminhada para o e-mail dos autores e o status do artigo fica como "Submetido".
Fluxo de eventos	1. O autor informa os dados do artigo. 2. São realizadas as validações, se os dados são válidos eles são armazenados.
Tratamento de exceções	1. Área do artigo vazia. (a) O sistema emite uma mensagem solicitando o preenchimento da área. 2. O arquivo não está no formato PDF. (a) O sistema emite uma mensagem informando que a extensão do arquivo está incorreta.

Tabela D.4: Distribuir artigo.

Nome do Caso de Uso	Distribuir artigo
Autor/Equipe	Gislaine Camila/Maringá
Data	12/03/2010
Atores	PresidenteComitêCientífico
Descrição	O ator PresidenteComiteCientifico encaminha os artigos aos revisores, considerando os temas selecionados.
Pré-condições	O prazo para submissão de artigos ter encerrado.
Pós-condições	É encaminhado um e-mail para os revisores e o status do artigo se torna "Em Revisão".
Fluxo de eventos	1. O PresidenteComitêCientífico disponibiliza os arquivos aos revisores para correção. 2. O status do artigo é alterado para "Em revisão".

Tabela D.5: Revisar artigo.

Nome do Caso de Uso	Revisar artigo
Autor/Equipe	Gislaine Camila/Maringá
Data	12/03/2010
Atores	Revisor
Descrição	O Revisor avalia o artigo segundo os critérios: relevância, conteúdo técnico e rigor científico, originalidade, qualidade da escrita e avaliação global.
Pré-condições	O artigo estar com o status "Em revisão"e dentro do prazo estabelecido para revisão.
Pós-condições	A revisão é cadastrada e o status do artigo passa a ser "Aguardando Parecer".
Fluxo de eventos	1. O revisor realiza a revisão para os artigos que estão disponíveis para ele. 2. Finaliza a revisão liberando o resultado para o PresidentedoComitê.
Tratamento de exceções	1. O campo Relevância está vazio. (a) O sistema emite uma mensagem solicitando o preenchimento da relevância. 2. O campo Conteúdo Técnico está vazio. (a) O sistema emite uma mensagem solicitando o preenchimento do campo. 3. O campo Rigor Científico está vazio. (a) O sistema emite uma mensagem solicitando o preenchimento do campo. 4. O campo Originalidade está vazio. (a) O sistema emite uma mensagem solicitando o preenchimento do campo. 5. O campo Qualidade da Escrita está vazio. (a) O sistema emite uma mensagem solicitando o preenchimento do campo. 6. O campo Avaliação Global está vazio. (a) O sistema emite uma mensagem solicitando o preenchimento do campo.

Tabela D.6: Consolidar revisão.

Nome do Caso de Uso	Consolidar revisão
Autor/Equipe	Gislaine Camila/Maringá
Data	13/03/2010
Atores	PresidenteComitêCientífico
Descrição	O PresidenteComitêCientífico realiza o acompanhamento das revisões, resolve os conflitos de revisões e encaminha a notificação da revisão aos autores.
Pré-condições	A data ser posterior a data limite para envio das revisões.
Pós-condições	A notificação das revisões é encaminhada aos autores e tem-se a lista de artigos aceitos.
Fluxo de eventos	1. O PresidenteComitêCientífico resolve os conflitos relacionados a revisões que não foram encaminhadas e a avaliação. 2. Encaminha notificação aos autores informando a data de submissão final para os artigos aceitos e os comentários dos revisores.
Tratamento de exceções	1. Revisões que não foram devolvidas. (a) O ator PresidenteComitêCientífico solicita as revisões e estabelece um novo prazo ou encaminha para outro revisor.

Apêndice

E

Documentação do Estudo de Viabilidade

Esse documento apresenta o memorial descritivo da abordagem, o questionário do perfil do participante e o questionário sobre a viabilidade da abordagem integrada de desenvolvimento e teste de software para equipes distribuídas.

E.1 Memorial Descritivo

Projeto de estudo de viabilidade
UMA ABORDAGEM INTEGRADA DE DESENVOLVIMENTO E TESTE DE SOFTWARE PARA EQUIPES DISTRIBUÍDAS
Dissertação de Mestrado
Gislaine Camila Lapasini Leal

Objetivo: analisar a viabilidade da abordagem proposta para oferecer apoio ao desenvolvimento distribuído de software com o propósito de adquirir conhecimento sobre a aplicação da mesma.

Abordagem: Com base em uma revisão de literatura e uma revisão sistemática foi possível evidenciar que há algumas lacunas no desenvolvimento distribuído de software (DDS) no que se refere ao processo de desenvolvimento. Desse modo, foram definidas

as atividades, os artefatos e os papéis necessários em cada etapa do desenvolvimento, considerando as peculiaridades do desenvolvimento com equipes distribuídas.

A abordagem proposta considera o processo de testes como um conjunto de tarefas à parte que pode ser aplicado ao longo do processo de desenvolvimento. A abordagem de desenvolvimento utiliza a (UML), por possuir informações relevantes para testes e estar em uso na academia e na maioria das indústrias de tecnologia, de modo que não há custos extras associados ao treinamento e integração no processo de desenvolvimento da aplicação.

Na abordagem de teste é utilizado o Perfil UML 2.0 (U2TP), que define uma linguagem para projetar, visualizar, especificar, analisar, construir e documentar os artefatos de teste de sistemas. Os artefatos de teste especificados pela abordagem são os recomendados pela norma IEEE 829, que descreve um conjunto básico de documentos de teste de software.

A visão geral da abordagem integrada, que mostra o fluxo de informações entre as Disciplinas, é apresentada na Figura E.1 . Os artefatos gerados são possíveis de refinamento nas Disciplinas posteriores.

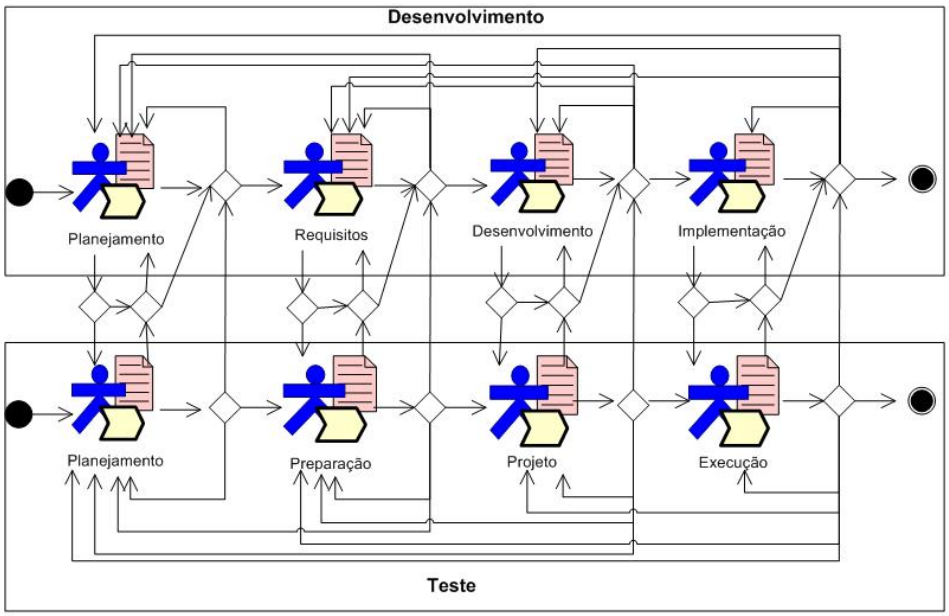

Figura E.1: Diagrama de Atividades da Abordagem Integrada de Desenvolvimento e Teste de Software.

A seguir são apresentados os elementos (papéis, artefatos e atividades) de cada uma das Disciplinas.

Planejamento - Desenvolvimento

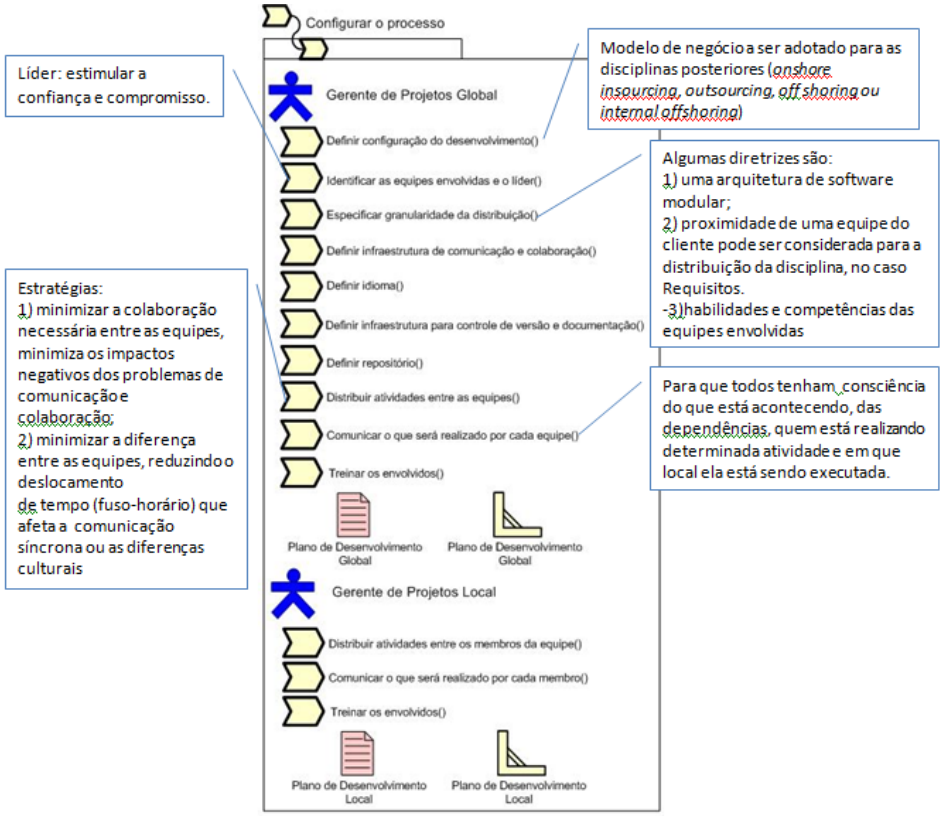

Figura E.2: Diagrama de Pacotes da Disciplina Planejamento.

Planejamento - Teste

Figura E.3: Diagrama de Pacotes da Disciplina Planejamento.

Requisitos - Desenvolvimento

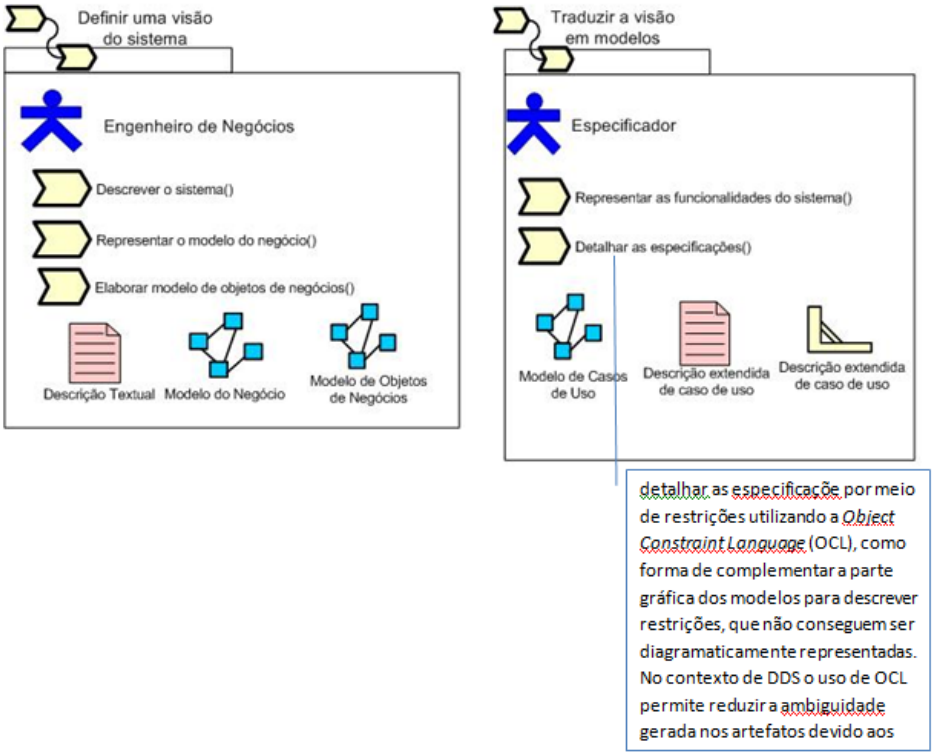

Figura E.4: Diagrama de Pacotes da Disciplina Requisitos.

Preparação - Teste

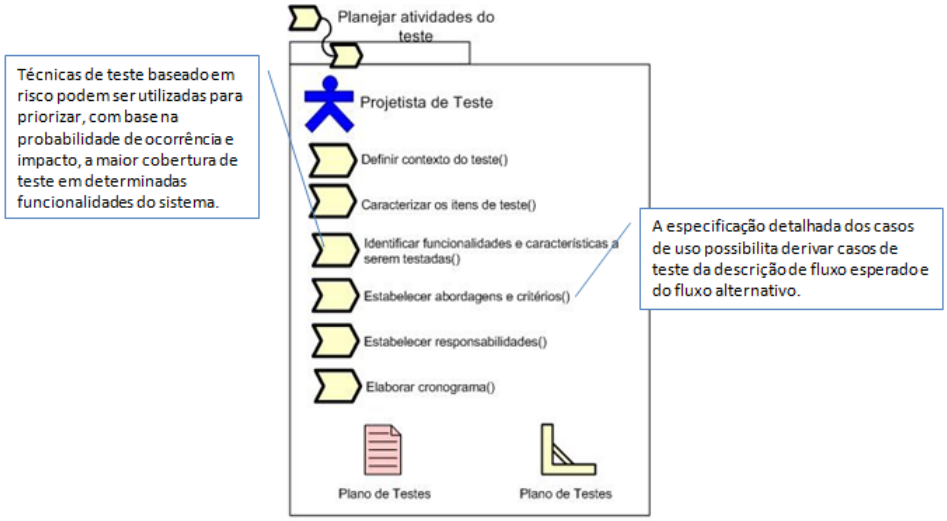

Figura E.5: Diagrama de Pacotes da Disciplina Preparação.

Desenvolvimento - Desenvolvimento

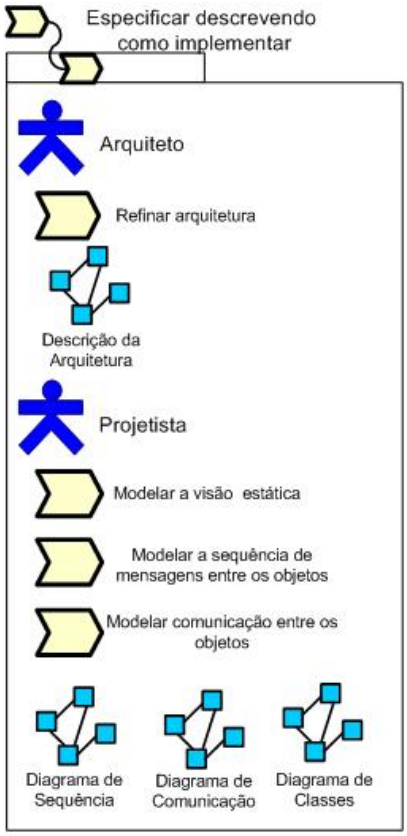

Figura E.6: Diagrama de Pacotes da Disciplina Desenvolvimento.

Projeto - Teste

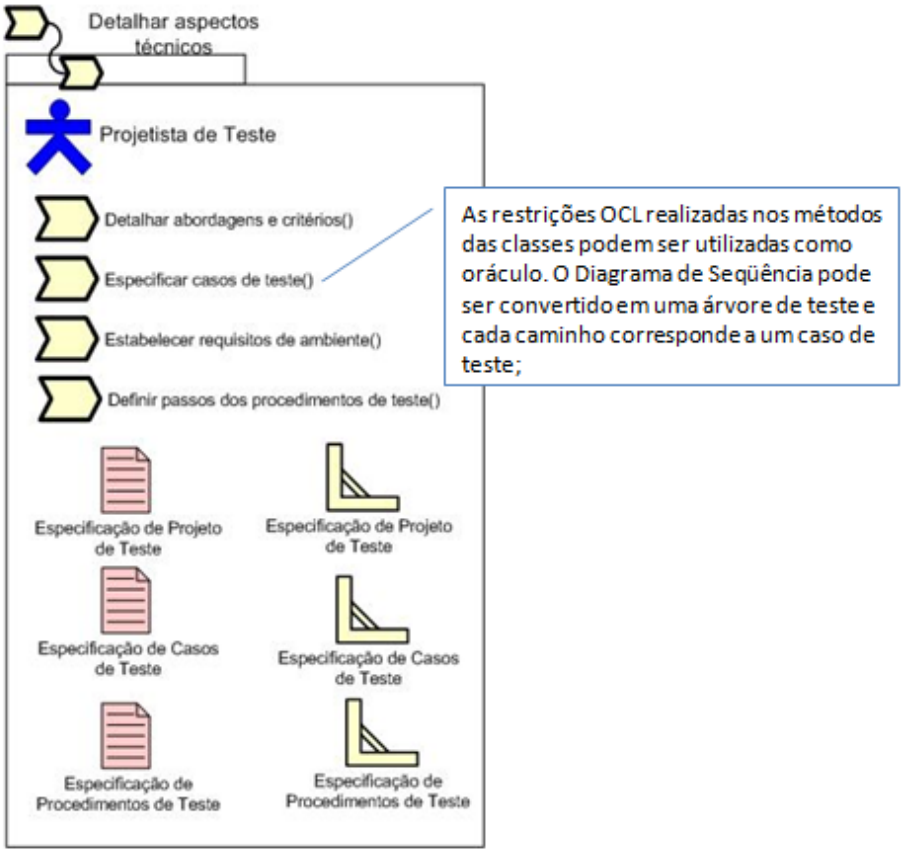

Figura E.7: Diagrama de Pacotes da Disciplina Projeto.

Implementação - Desenvolvimento

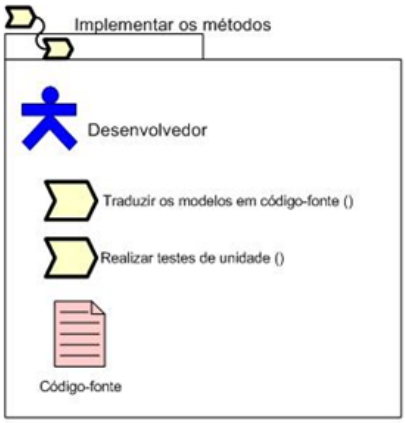

Figura E.8: Diagrama de Pacotes da Disciplina Implementação.

Execução - Teste

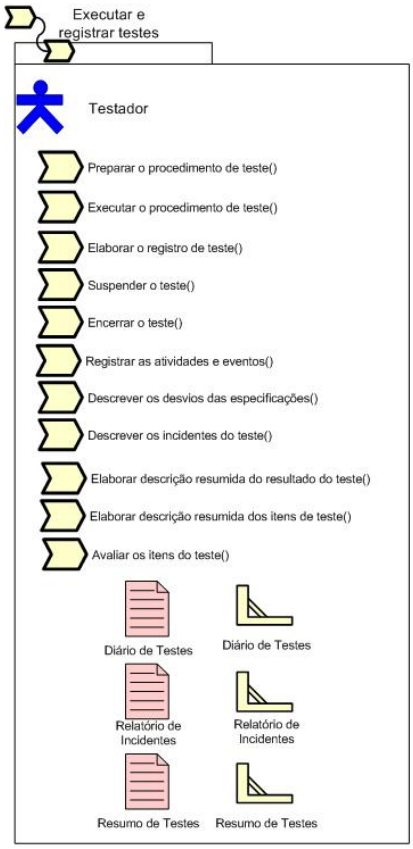

Figura E.9: Diagrama de Pacotes da Disciplina Execução.

E.2 Questionários

Nesta Seção são apresentados os questionários utilizados na instrumentação do estudo conduzido.

E.2.1 Questionário 1 - Perfil do Entrevistado

Nome:_____

Instituição:_____

1. Qual o seu grau de formação?

( ) Graduado ( ) Especialista ( ) Mestrando ( ) Mestre ( ) Doutorando ( ) Doutor

2. Qual a sua experiência com desenvolvimento de sistemas?

( ) Nunca desenvolvi ( ) Desenvolvi para uso pessoal
( ) Desenvolvi como partes de trabalho de curso (___ projetos)
( ) Desenvolvi software em ambiente industrial (___ anos)

3. Tem conhecimento em desenvolvimento distribuído de software?
( ) Nenhum ( ) Básico ( ) Intermediário ( ) Avançado

4. Qual a sua experiência com desenvolvimento distribuído de software?
( ) Acadêmica ( ) Industrial ( ) Ambas

5. Tem experiência em gerência de projetos?
( ) Nenhum ( ) Básico ( ) Intermediário ( ) Avançado

6. Tem conhecimento em teste de software?
( ) Nenhum ( ) Básico ( ) Intermediário ( ) Avançado

E.2.2 Questionário 2 - Estudo de Viabilidade

1. Em que grau o conjunto de disciplinas apresentado é satisfatório para o desenvolvimento distribuído?
( ) Nenhum ( ) Baixo ( ) Intermediário ( ) Satisfatório

2. Em que grau as atividades apresentadas atendem às necessidades do desenvolvimento distribuído?
( ) Nenhum ( ) Baixo ( ) Intermediário ( ) Satisfatório

3. Em que grau os artefatos apresentados atendem às necessidades do desenvolvimento distribuído?
( ) Nenhum ( ) Baixo ( ) Intermediário ( ) Satisfatório

4. Em que grau o uso de uma notação (UML) que oferece informações relevantes para as duas equipes (desenvolvimento e teste) pode amenizar os problemas de comunicação?
( ) Nenhum ( ) Baixo ( ) Intermediário ( ) Satisfatório

5. Em que grau a identificação das equipes, bem como de seus membros e responsabilidades através dos Planos de Desenvolvimento Global, Plano de Desenvolvimento Local e

Plano de Testes podem aumentar a percepção de um projeto?
( ) Nenhum ( ) Baixo ( ) Intermediário ( ) Satisfatório

6. A definição da abordagem oferece uma nomenclatura comum a todos os envolvidos no projeto e possibilita entender a dependência das atividades e artefatos. Em que grau a abordagem apresentada atende às necessidades do DDS?
( ) Nenhum ( ) Baixo ( ) Intermediário ( ) Satisfatório

7. Em que grau o uso OCL (Object Constraint Language) para especificar restrições pode reduzir os problemas de ambigüidade nos artefatos (diagrama de classes)?
( ) Nenhum ( ) Baixo ( ) Intermediário ( ) Satisfatório

8. Em que grau a padronização dos artefatos e atividades, definidos na abordagem integrada, podem impactar na qualidade dos produtos desenvolvidos?
( ) Nenhum ( ) Baixo ( ) Intermediário ( ) Satisfatório

9. Em que grau as atividades de teste ao longo de todas as disciplinas de desenvolvimento podem reduzir os problemas de integração?
( ) Nenhum ( ) Baixo ( ) Intermediário ( ) Satisfatório

10. Você visualiza atividades e artefatos adicionais que devem ser contempladas pela abordagem? Se sim, quais?
( ) Sim ( ) Não
