Evandro Moreira Pinto

Proposta de um processo de gerenciamento das interações de uma organização de desenvolvimento de software com uma ou mais fábricas de software em projetos que utilizam o RUP como modelo de processo

Dissertação apresentada ao Instituto de Pesquisas Tecnológicas do Estado de São Paulo – IPT como requisito para a obtenção do título de mestre em Engenharia da Computação.

Área de concentração: Engenharia de Software

Orientador: Prof. Dr. Ivanir Costa

São Paulo

Março/2008

Ficha Catalográfica
Elaborada pelo Departamento de Acervo e Informação Tecnológica – DAIT
do Instituto de Pesquisas Tecnológicas do Estado de São Paulo - IPT

P659p Pinto, Evandro Moreira
Proposta de um processo de gerenciamento das interações de uma organização de desenvolvimento de software com uma ou mais fábricas de software em projetos que utilizam o RUP como modelo de processo. / Evandro Moreira Pinto. São Paulo, 2008.
197p.

Dissertação (Mestrado em Engenharia de Computação) - Instituto de Pesquisas Tecnológicas do Estado de São Paulo. Área de concentração: Engenharia de Software.

Orientador: Prof. Dr. Ivanir Costa

1. Engenharia de software 2. Rational Unified Process - RUP 3. Modelo de processo 4. Unified Modeling Language - UML 5. Tese I. Instituto de Pesquisas Tecnológicas do Estado de São Paulo. Coordenadoria de Ensino Tecnológico II. Título

08-112 CDU 004.41(043)

Tudo aquilo que insistimos em fazer se torna fácil, não porque a natureza da tarefa mude, mas porque nossa capacidade aumenta.

Herber J. Grantt.

Nenhum sucesso profissional compensa o fracasso no lar

David O. McKay

Dedicatória

Dedico este trabalho à minha amada esposa Nereida e a meus amados filhos, Wagner, Marcos e Marcelo, que me apoiaram com muita compreensão e incentivo em todos os momentos deste trabalho.

Ao meu querido pai Antônio Teixeira Pinto, que tanto se esforçou para que eu obtivesse toda a educação necessária em minha vida. Tenho certeza de que ele estaria compartilhando intensamente da felicidade que estou vivenciando na obtenção deste tão sonhado título de Mestre.

Agradecimentos

Agradeço ao Prof. Dr. Ivanir Costa, que me orientou com seu grande conhecimento e longa experiência em Engenharia de Software. Agradeço também ao Prof. Dr. Mário Yoshikasu Miyake (IPT), pelo seu incentivo nos momentos difíceis que enfrentei durante a elaboração desta dissertação. Estou grato também à Professora Ivana Fátima dos Santos Tavares, pela preciosa revisão gramatical em todo o texto desta dissertação.

Resumo

O gerenciamento de projetos em organizações de desenvolvimento de software com uso do modelo de processo seqüencial clássico e com subcontratação de fábrica de software, tem sido assunto de estudos acadêmicos e largamente praticado no mercado brasileiro e internacional. No que se refere ao uso do *Rational Unified Process* (RUP) como modelo de processo, a literatura também tem uma boa cobertura mas, unicamente do ponto de vista da fábrica de software. No que se refere ao uso do RUP em organizações de desenvolvimento de sistemas subcontratantes, a literatura é bastante escassa.

Este trabalho tem por objetivo propor mecanismos de controle das interações entre uma organização de desenvolvimento de sistemas e uma ou mais fábricas de software subcontratadas para participar de um projeto incremental e iterativo, usando o RUP como modelo de processo de software e a *Unified Modeling Language* (UML) como linguagem de notação. Responde perguntas tais como: Como definir os escopos de atuação da organização subcontratante e da fábrica de software subcontratada? Como controlar as interações entre essas duas organizações usando processo incremental e iterativo? A UML resolve problemas de comunicação entre equipes que trabalham de forma complementar, mas separadas geograficamente?

O resultado obtido na pesquisa foi a diminuição do acoplamento da fábrica de software em relação à organização subcontratante, eliminando a necessidade de sucessivas reuniões entre as duas organizações para esclarecer ambigüidades, inconsistências, falta de clareza e completeza das especificações entregues como insumos para a fábrica de software. Com isso, passa a ser possível uma organização de desenvolvimento de software subcontratar fábricas de software geograficamente distantes em busca de qualidade, produtividade e menores custos.

Palavras-chave: Engenharia de software, fábrica de software, gestão de projetos, processo de software, *Rational Unified Process* – RUP, subcontratação, *Unified Modeling Language* – UML.

Abstract

Proposal of a process for interaction management of a software development organization with one or more software factories in projects which utilize the RUP as a process model

The project management in software developing organization, using the classic sequential process and sub-contraction of software factory, has been the subject of academic studies and widely practiced in the international as well as in the Brazilian market. In reference to the Rational Unified Process (RUP), as a process model, the literature is well defined, but only from the point of view of the software factory. But using RUP at the software developing organization, the literature is very poor.

The purpose of this work is to propose a controlling mechanism of interactions between a system development organization and one or more software factories sub-contracted, participating on an iterative and incremental project using Rational Unified Process (RUP) as a process model and the Unified Modeling Language (UML) as a notation language. Answering questions such as: How to define the scope of the work of the system development organization sub-contractor and the software factory sub-contracted? How to control the interaction between these two organizations by using the iterative and incremental process? Does the UML solve communication problems between teams working complementing each other but geographically remote?

The obtained result in this research was the coupling reduction of the software factory from the system development organization sub-contractor, eliminating the need for repeated meetings between the two to clarify ambiguities, inconsistencies, absence of clarification, and completion of the specifications given to the software factory as input. This allows a software development organization to sub-contract geographically remote software factories for the pursuit of quality, productivity, and lesser costs.

Keywords: Software engineering, software factory, project management, software process, Rational Unified Process – RUP, sub-contraction, Unified Modeling Language – UML.

Lista de Figuras

Lista de Tabelas

Lista de Siglas

SIGLA	**Descrição**
CASE	*Computer Aided Software Engineer*
DDSBE	Departamento de Desenvolvimento de Sistemas de Bilhetagem Eletrônica
EA	*Enterprise Architecture*
EBE	Empresa de Bilhetagem Eletrônica
ECFS	Empresa Contratante da Fábrica de Software.
FS	Fábrica de Software
ODS	Organização de Desenvolvimento de Sistemas
OSI	Ordem de Serviço Interno
RUP	*Rational Unified Process*
SBE	Sistema de Bilhetagem Eletrônica
SS	Solicitação de Serviço
UML	*Unified Modeling Language*

Sumário

Capítulo 1 – Introdução

Um das responsabilidades do gerente de projetos de software de uma Organização de Desenvolvimento de Sistemas (ODS)[1] que pretende subcontratar[2] uma Fábrica de Software (FS) para a implementação de um sistema é definir quais as atividades que serão desenvolvidas internamente na ODS e quais os artefatos que deverão ser suficientemente criados para que uma FS possa produzir um software com qualidade, sem a necessidade de constantes reuniões entre a ODS e a FS com a finalidade de tirar dúvidas originárias de especificações inconsistentes, ambíguas e incompletas, de fazer correções e de complementar o levantamento de requisitos.

O modelo de processo seqüencial clássico facilita a utilização de uma FS porque o controle do processo só passa de uma fase para a fase subseqüente, quando todas as atividades da fase anterior forem concluídas [LEF2000], [BOE1988], [PRE1995]. A expectativa é que as fases desenvolvidas na FS sigam um processo seqüencial, sem a necessidade da reexecução das atividades das fases anteriores já concluídas pela ODS. Dessa forma, fica bem mais fácil a gerência do projeto definir o "ponto de corte" entre as atividades da ODS e da FS.

---

[1] Neste trabalho, o termo ODS pode ser aplicado a:

- um departamento de desenvolvimento de sistemas dentro de uma empresa consumidora de tecnologia, a exemplo de um banco, de uma fábrica de automóveis ou de uma empresa de serviço, que utiliza uma FS para a implementação do sistema;
- uma consultoria em desenvolvimento de sistemas contratada por uma empresa consumidora de tecnologia para desenvolver um determinado sistema de informação desde as fases iniciais, porém usando o serviço de uma FS para a implementação do sistema [CES2003].

[2] O termo "subcontratação" usado no presente trabalho é baseado no conceito usado pelo CMM [FIO1998].

A definição deste “ponto de corte” é bem mais difícil quando se deseja utilizar um processo incremental e iterativo, tal como o *Rational Unified Process* (RUP), como processo de desenvolvimento de sistemas com uso de FS. O RUP é um processo incremental, iterativo, dirigido por casos de uso e centrado em arquitetura [KRU2001]. Como demonstrado na Figura 1, a iteratividade do processo leva à repetição da execução de disciplinas em todas as iterações nas quatro fases do RUP.

Quando se trata de um projeto com uma única equipe, sem a utilização de FS, todas as disciplinas envolvidas nas iterações das quatro fases do RUP são executadas unicamente pela ODS. Isso facilita a comunicação entre o gerente do projeto e toda a sua equipe, facilitando o controle do processo, pois todos se encontram no mesmo ambiente de trabalho.

Contudo, se o projeto previr o uso de uma FS, o ciclo de desenvolvimento terá parte desenvolvida pela ODS e outra parte, pela FS. Como, ao longo do ciclo de vida de um projeto que utiliza o RUP existem quatro fases e cada fase terá, no mínimo, uma iteração, o controle do processo se torna mais complexo, porque, em cada iteração, a FS será solicitada para desenvolver diferentes demandas. Em um processo seqüencial clássico, a FS seria solicitada uma única vez, mas, em um processo incremental e iterativo, a FS será solicitada em cada iteração, conforme será demonstrado no Capítulo 5 deste trabalho. É válido lembrar que, internamente, a FS poder utilizar qualquer outro processo, inclusive o seqüencial, para atender à demanda.

Conforme demonstrado na Figura 1, com exceção da disciplina de Implantação, todas as demais já são iniciadas na fase de Iniciação e se repetem a cada iteração ao longo das quatro fases do RUP. Conforme o planejamento, a FS já poderá ser envolvida na primeira iteração, repetindo-se em cada iteração até o fim da fase de Transição.

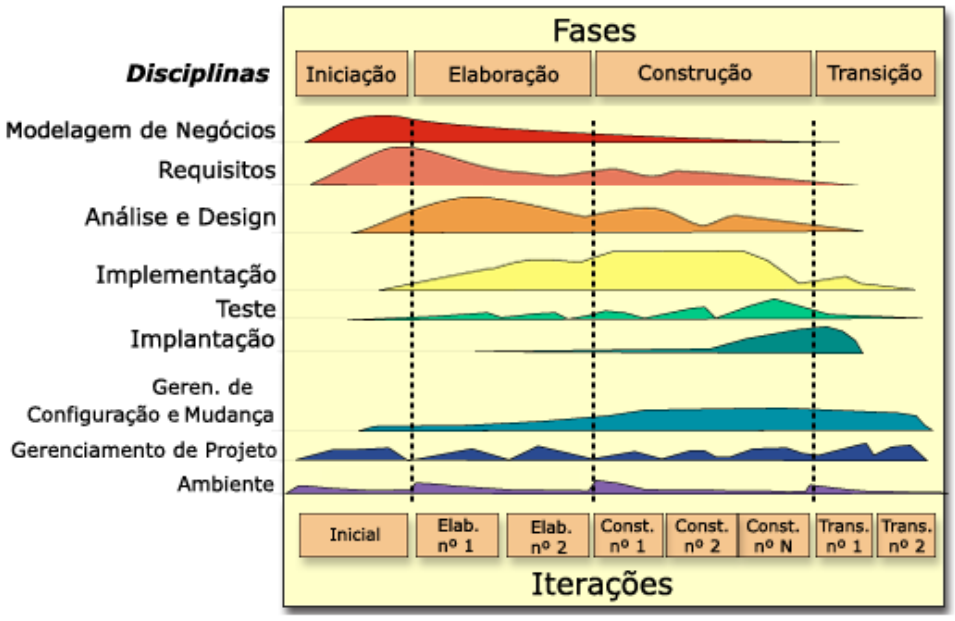


Figura 1 - Visão geral do RUP [RUP2002]

Se uma ODS desejar executar as disciplinas de Modelagem do Negócio e Requisitos e fazer uso da FS para executar as disciplinas de Análise-*Design* e a Implementação em cada iteração, ela necessitará interagir com a FS em cada iteração. Assim, a entrega das especificações para a FS é feita de forma parcelada ou evolutiva, a cada iteração. Diferentemente, no modelo de processo seqüencial clássico, devido à não iteratividade, a FS receberá as especificações inteiramente de uma única vez.

1.1 Problema

O problema da utilização do RUP em um projeto que usa FS reside na iteratividade, fato que eleva a complexidade do gerenciamento do projeto porque parte das atividades de cada iteração é feita na ODS e uma outra parte na FS.

Devido ao fato de o RUP ser um modelo de processo usado como referência para a definição de um processo efetivo de desenvolvimento de sistemas, ele não aborda aspectos relacionados à subcontratação de FS, nem como e nem

quando ela deve ser acionada pela ODS. Isso dificulta a definição de quais artefatos devem compor o pacote de especificações que deve ser produzido pela ODS e envidado para a fábrica, sem o risco de estar incompleto.

Especificações incompletas, ambíguas, redundantes e inconsistentes sem uma notação exata e clara, adicionadas à elevada complexidade no gerenciamento das execuções das iterações, resultam numa elevada dependência da FS com relação à ODS. Essa dependência se reflete na necessidade de freqüentes reuniões entre a ODS e FS para complementar e corrigir os requisitos, modelos e as especificações recebidas da ODS, estendendo os prazos e elevando os custos do projeto, além de elevar os riscos de se produzir um sistema que não atenda às expectativas dos usuários e dos patrocinadores.

Uma análise mais cuidadosa dos problemas descritos acima mostra claramente que existem duas causas-raíz. A primeira é a ausência de mecanismos que auxiliem no controle das interações entre a ODS e FS em cada iteração do processo de software. A segunda é o acoplamento da FS à ODS.

Uma FS atuará de forma acoplada à ODS quando houver um elevado grau de dependência daquela em relação a esta, por não se conseguir desenvolver o produto solicitado a partir dos artefatos recebidos, sendo, pois, necessária a presença de profissionais da FS dentro da ODS, ou o contrário, para que haja uma complementação das especificações dos artefatos entregues.

É válido salientar que esse acoplamento não está relacionado à necessidade de os profissionais da FS interagirem diretamente com os usuários da empresa subcontratante, dona da ODS, para fazer o levantamento de requisitos para a produção dos artefatos necessários. Isso é plenamente possível e adequado quando a ODS, por algum motivo, subcontrata uma FS para realizar esse levantamento.

O acoplamento se faz existir quando a FS não consegue elaborar o produto solicitado a partir dos artefatos entregues pela ODS por motivos de falhas na notação, especificações incompletas, inconsistentes e ambíguas. Essas falhas provocam dependência da FS em relação à ODS para fazer as correções necessárias. Em alguns casos, essa dependência é tamanha que requer que a FS envie um grupo de profissionais para trabalharem diretamente dentro da ODS, elevando o grau de acoplamento da FS em relação à ODS, impedindo que esta, lance mão de fábricas situadas em localidades distantes em busca de qualidade, especialização e redução de custos.

Alguns trabalhos [MRQ2004-A], [MRQ2004-B], [CAB2006], [COS2003] abordam pesquisas que tratam da utilização de FS com uso do RUP cobrindo todo o ciclo de vida do projeto, em que todas as disciplinas de todas as iterações são desenvolvidas na própria FS sem o controle da ODS. Esses trabalhos não especificam como se faz o gerenciamento do projeto do ponto de vista da ODS.

Não foi encontrado no levantamento bibliográfico nenhum artigo acadêmico-científico que versasse sobre o uso de um processo incremental e iterativo de desenvolvimento de software, com uso de uma ou de mais FS, porém com o controle do processo feito ativamente pela ODS. Entretanto, dentre os artigos não acadêmicos levantados na pesquisa bibliográfica, só foi identificado um único *white paper*[3] [MEL2001] que trata desse tema.

Mello [MEL2001] propõe um processo de subcontratação que pode ser total ou parcial da execução do processo de software. Propõe ainda seis tipos de cenários envolvendo o fornecedor ao longo das iterações do processo de software. Esses cenários prevêem somente um fornecedor e já define todo o escopo de atuação do fornecedor por meio do cronograma de fornecimento que fica no controle da ODS e do cronograma de fabricação que fica sob o controle

[3] *White Paper* são trabalhos publicados por alguma instituição, porém sem os rigores característico de um trabalho científico.

do fornecedor. Mello sugere que a ODS deve diligenciar o sincronismo entre os dois cronogramas, mas não detalha como isso deverá ser feito. É válido observar que o fato de o cronograma de fabricação encontrar-se no poder do fornecedor sugere que o controle da fabricação fique nas mãos deste. Por isso, pode-se afirmar que, no processo de subcontratação proposto por Mello, a ODS exerce um fraco controle do processo incremental e iterativo de desenvolvimento de software.

Os problemas acima descritos sugerem as seguintes perguntas:

- Dentro de uma iteração, quais os pontos em que uma ODS poderá acionar uma FS?
- Quais os artefatos que a ODS deverá produzir em uma iteração para serem enviados à FS?
- Quais são os artefatos, e suas respectivas características, necessários à FS para que esta possa atuar de forma desacoplada da ODS em uma iteração?
- Qual o processo que deverá ser utilizado pela ODS no gerenciamento das interações com uma ou mais FS subcontratadas para atuarem em um processo de software incremental e iterativo?

Todo o desenvolvimento do presente trabalho baseia-se nestas questões, a partir das quais as hipóteses descritas na seção seguinte foram formuladas [GIL2006].

1.2 Hipóteses

Segundo Gil [GIL2006], uma hipótese é uma expressão verbal susceptível de ser declarada verdadeira ou falsa. É uma proposição testável que pode vir a ser a solução do problema identificado. Devido ao fato de que cada pergunta acima descrita ter sido formulada a partir dos problemas identificados, as hipóteses que serão comprovadas no presente trabalho foram definidas a partir dessas perguntas.

Hipótese-1

Pergunta-problema:

Dentro de uma iteração, quais os pontos em que uma ODS poderá acionar uma FS?

Enunciado da hipótese-1:

Os pontos, dentro do fluxo de uma iteração, nos quais a ODS deverá acionar a FS dependem do cenário[4] estabelecido para a iteração em foco, em que cada uma dessas organizações deverá ter um escopo[5] de atuação executando atividades pertencentes às disciplinas operacionais[6] do RUP.

Hipótese-2

Pergunta-problema:

Quais os artefatos que a ODS deverá produzir em uma iteração para serem enviados à FS?

Enunciado da hipótese-2:

Os artefatos que a ODS deverá produzir em uma iteração para serem enviados para a FS dependem do escopo de atuação da ODS e da(s) FS(s) envolvida(s) na iteração em foco.

---

[4] Cenário é um conjunto de atividades que deverão ser executadas pela ODS e pelas FS envolvidas em uma iteração. Um cenário é formado pela soma dos escopos da ODS e das FS envolvidas em uma iteração. O Capítulo 5 apresenta um modelo dos cenários e escopos em uma iteração.

[5] Escopo é o conjunto de atividades que são executadas pela ODS ou por uma FS por ocasião de uma iteração.

[6] Disciplina operacional: no presente projeto o temo "disciplina operacional" significa as seguintes disciplinas do RUP: Modelagem do Negócio, Requisitos, Análise-Design, Implementação, Teste e Implantação.

Hipótese-3

Pergunta-problema:

Quais são os artefatos, e suas respectivas características, necessários à FS para que esta possa atuar de forma desacoplada da ODS em uma iteração?

Enunciado da hipótese-3:

Os artefatos necessários para que uma FS possa gerar o produto solicitado em uma iteração e o faça de forma desacoplada da ODS, depende do escopo da FS na iteração em foco, da correta utilização da UML na especificação desses artefatos e do pleno conhecimento em UML por parte dos profissionais da FS envolvidos no desenvolvimento dos produtos solicitados.

Hipótese-4

Pergunta-Problema:

Qual o processo que deverá ser utilizado pela ODS no gerenciamento das interações com uma ou mais FS subcontratadas para aturem em um processo de software incremental e iterativo?

Enunciado da hipótese 4:

Para que a ODS possa controlar as interações com uma ou mais FSs envolvidas no cenário de cada iteração, é necessário um processo que execute o planejamento dos cenários e respectivos escopos da ODS e de cada FS envolvida na iteração e controle a execução das disciplinas operacionais do RUP dentro dos escopos da ODS e das FS envolvidas.

1.3 Objetivo

O objetivo do presente trabalho é propor mecanismos de controle entre a ODS e uma ou mais FSs subcontratadas para a produção de um sistema desenvolvido de forma incremental e iterativa, usando o RUP como modelo de processo de

software e a UML como linguagem de notação, eliminando a necessidade de repetidas reuniões entre as duas organizações para esclarecer ambigüidades, inconsistências, falta de clareza e completeza das especificações entregues como insumos para a FS.

A busca pela eliminação dessas reuniões não significa a busca pela completa segregação da FS em relação à ODS. O foco deste trabalho está voltado para a obtenção da interdependência entre as duas organizações na ocasião da produção dos artefatos criados pelas disciplinas operacionais do RUP. Essa interdependência deverá ser efetivada pela ausência da necessidade das reuniões para complementação e esclarecimento de dúvidas sobre os artefatos entregues como insumos para as FSs envolvidas.

Não será objeto do presente trabalho a busca pela eliminação ou redução de reuniões entre aquelas organizações, por ocasião da execução das disciplinas gerenciais[7] do RUP.

O ponto focal do presente trabalho é a ODS, e não a FS. Portanto, neste contexto, a FS é "caixa preta", não interessando a definição dos processos usados para a produção dos artefatos solicitados pela ODS.

É válido salientar que, embora o presente trabalho tenha como pressuposto que a ODS detenha o controle do processo de desenvolvimento, o trabalho não irá interferir nos processos de cada disciplina do RUP. A execução de cada disciplina também será considerada "caixa preta" nos mecanismos a serem definidos. O foco do processo que será proposto está na ativação de cada disciplina do RUP dentro da ODS ou na FS.

---

[7] O termo "disciplina gerencial" no presente trabalho compreenderá as seguintes disciplinas do RUP: Gestão de Configuração e Mudança, Gestão do Projeto e Ambiente.

Os mecanismos de controle que serão propostos deverão se submeter às restrições de ordem prática e econômica, porque a intenção desta pesquisa é propor algo factível do ponto de vista da ODS e da FS.

A busca pela utilização do RUP e da UML, justifica-se por ser respectivamente, um modelo de processo de software e uma linguagem de modelagem de softwares bastante maduros, que cobrem todo o ciclo de vida do software, permite a produção de uma boa especificação e é fartamente utilizada mundialmente [MRQ2004-B].

1.4 Motivação

A motivação dessa proposta está baseada nos seguintes fatos:

- Necessidade de definir mecanismos para contribuir com a redução da dependência da FS em relação à ODS, evitando reuniões adicionais entre essas duas organizações para complementar os requisitos e tirar dúvidas sobre o negócio. Com isso não será mais necessária a permanência de profissionais da FS dentro da ODS, nem da ODS dentro da FS. A FS passa a ser uma prestadora de serviço totalmente desacoplada, permitindo uma rápida substituição de uma FS por outra que venha oferecer serviços de melhor qualidade, menor custo e melhores prazos.
- A eliminação dessa dependência irá permitir que a ODS possa subcontratar FSs localizadas em regiões distantes.
- Os profissionais da ODS terão mais tempo para se dedicar a atividades relacionadas ao gerenciamento dos projetos, dos requisitos, da qualidade, das configurações, das mudanças e do ambiente.

1.5 Organização do trabalho

O presente trabalho está estruturado da seguinte forma:

- Capítulo 1: Faz a apresentação do problema, define as hipóteses e declara o tema, objetivos, motivação e a metodologia usada no trabalho.
- Capítulo 2: Descreve a metodologia utilizada.
- Capítulo 3: Descreve o RUP como modelo de processo de software.
- Capítulo 4: O conceito de fábrica de software utilizado no presente trabalho.
- Capítulo 5: Definição dos mecanismos de controle das interações entre a ODS e FS.
- Capítulo 6: Apresentação da pesquisa-ação com uma completa narrativa de todos os fatos ocorridos por ocasião do desenvolvimento do projeto usado como objeto de estudo. É apresentado o histórico de adaptações dos mecanismos definidos no Capítulo 5 em busca do aperfeiçoamento desses mecanismos para que possam também ser usados em outros projetos e em outras instalações.
- Capítulo 7: Conclusões.
- Capítulo 8: Bibliografia.
- Anexos.

Capítulo 2 – Metodologia

A metodologia usada no presente trabalho foi uma adaptação do processo indicado por [CIR2003] que consiste dos seguintes passos:

1. Seleção e delimitação do tema.
   O tema foi definido através da percepção da conjunção dos seguintes fatos:
   a. crescente tendência da utilização de FS no desenvolvimento de sistemas;
   b. elevada dependência das FS em relação à ODS;
   c. necessidade de utilização de processos incrementais e iterativos, tais como o RUP, em projetos de grande porte e de elevada complexidade;
   d. elevada utilização da UML como notação de projetos de software padronizando a comunicação entre diferentes equipes de projeto dentro do seu ciclo de vida.

   Buscou-se a definição do problema e suas causas-raiz a fim de definir hipóteses para guiar a pesquisa.

2. Levantamento da bibliografia sobre Processo de Software.
   Um levantamento preliminar da bibliografia permitiu a visualização de lacunas no conhecimento científico no que se refere aos processos de gerenciamento de projetos de software com uso de FS, porém de forma incremental e iterativa. A pesquisa bibliográfica identificou uma grande quantidade de artigos acadêmicos e *whitepapers* versando sobre processos de software usados em FS, porém somente um trabalho [MEL2001] foi encontrado tratando a interação entre a ODS e FS com uso do RUP como modelo de processo.

3. Definição das hipóteses para a solução do problema.

   As hipóteses foram mapeadas a partir dos problemas apresentados no Capítulo 1. Essas hipóteses tiveram o papel de guia em todo o desenrolar do projeto de pesquisa em busca da comprovação das suas veracidades.

4. Definição da estrutura do trabalho.

   Trata-se de um conjunto de capítulos, seções e subseções que formam a estrutura básica da dissertação. Isso facilitou a montagem de idéias, apontamentos e a distribuição de todas as citações bibliográficas que se conseguiu na medida do avançamento da pesquisa. Ajudou também a definir quais fontes bibliográficas deveriam ser usadas diante de um elevado número de fontes que foram acessadas no levantamento preliminar. Com isso a dissertação já foi sendo montada de forma organizada e estruturada. Contudo, essa estrutura sofreu freqüentes modificações na medida em que se avançou na pesquisa devido à necessidade de adequá-la às novas idéias e definições que precisavam ser inseridas.

5. Definição da modalidade de pesquisa a ser realizada.

   Para que as hipóteses definidas fossem comprovadas, foi necessária a realização de um projeto de pesquisa. A correta definição da modalidade de pesquisa se tornou um passo crítico para o sucesso do presente projeto, pois uma escolha inadequada poderia impedir ou dificultar todo o trabalho de comprovação das hipóteses. A modalidade escolhida foi "Pesquisa-Ação". Trata-se de pesquisa experimental proposta por Thiollent, 1985 apud Gil [GIL2006], cuja definição é:*"um tipo de pesquisa com base empírica que é concebida e realizada em estreita associação com uma ação ou com a resolução de um problema coletivo e no qual os pesquisadores e participantes representativos da situação ou do problema estão envolvidos de modo cooperativo ou participativo".*

A razão pela qual foi utilizado este modelo de pesquisa foi a necessidade de uma oportunidade para o desenvolvimento e refinamento dos mecanismos, objeto da pesquisa.

Para a comprovação das hipóteses, foi definido um conjunto de mecanismos gerenciais e operacionais, tais como: processos, regras, normas*, templates* e base de dados a serem utilizados pela ODS e pela FS de forma que pudessem ser planejados, aplicados, avaliados e, principalmente, refinados ao longo da pesquisa, até que o objetivo do projeto fosse alcançado. Para que isso pudesse ocorrer, era necessária uma modalidade de pesquisa que permitisse essa interação com os resultados para fazer os necessários ajustes dos mecanismos ao longo do processo da pesquisa para a obtenção do resultado esperado.

6. Escolha do projeto a ser usado na pesquisa-ação.
   Foi escolhida uma empresa de consultoria em sistemas de informação com sede em São Paulo, que presta serviços de desenvolvimento de sistemas sob encomenda. O projeto escolhido é de um cliente sediado no Rio de Janeiro. O Capítulo 6 descreve em detalhes os motivos que levaram à escolha desse projeto.

7. O planejamento e execução da pesquisa experimental seguiram os seguintes passos:
   a. Planejamento do processo de desenvolvimento do sistema (metodologia) a partir do RUP. Trata-se de uma instância do RUP pertencente ao nível M0 dentre os quatro níveis de modelagem conforme sugerido pelo SPEM [OMG2005]. É válido observar que, embora não seja foco deste trabalho, este planejamento se fez necessário devido ao fato de que o sistema precisava ser desenvolvido com uso de um processo derivado do RUP.

b. Definição do processo de controle das interações entre a ODS e FS (veja Figura 12). Esse processo é o objeto de teste relacionado à hipótese de número quatro do presente trabalho.
c. Desenvolvimento dos templates de documentos usados no processo, tais como Solicitação de Serviço (SS), para ser usada como instrumento de comunicação entre a ODS e a FS e Ordem de Serviço Interna (OSI) para ser usada no controle das interações com as unidades organizacionais internas na ODS.
d. Definição de normas e regras internas de procedimentos a serem usadas na ODS.
e. Contratação da equipe do projeto, tanto para a ODS como para a FS.
f. Desenvolvimento do treinamento em UML e no processo a ser seguido no projeto.
g. Treinamento das equipes da ODS e da FS.
h. Execução do processo da disciplina de gestão de projetos proposto pelo RUP conforme demonstrado na figura 11. Note que a atividade “Gestão da Iteração” é a responsável pela execução do processo de controle das interações entre a ODS e a(s) FS(s) envolvidas.
i. Aplicação dos mecanismos definidos na ODS e na FS.
j. Registro de dados por ocasião da aplicação dos mecanismos.
k. Análise dos dados coletados.
l. Ajustes dos mecanismos gerenciais e operacionais para uma reaplicação na ODS e na FS.
m. Este ciclo se repetiu até que os resultados esperados foram alcançados.

8. Tabulação dos dados obtidos na pesquisa-ação.

9. Elaboração da dissertação.

Os passos acima descritos não foram necessariamente executados nesta seqüência. Em várias ocasiões, alguns passos foram realizados paralelamente e, em ocasiões necessárias, alguns deles foram repetidos em busca de melhoramentos.

2.1 Comprovação das Hipóteses

As hipóteses serão comprovadas da seguinte maneira:

Hipótese-1:

Os pontos, dentro do fluxo de uma iteração, nos quais a ODS deverá acionar a FS dependem do cenário estabelecido para a iteração em foco, onde cada uma dessas organizações deverá ter um escopo de atuação executando atividades pertencentes às disciplinas operacionais do RUP.

Como comprovar a hipótese-1:

Para comprovar essa hipótese se faz necessário o estabelecimento de:

- Todas as possibilidades de escopos, onde cada um é constituído por uma ou por mais disciplinas do RUP.
- Todas as possibilidades de cenários, onde cada um é um arranjo de escopos executados pela ODS e pela FS ou FSs envolvidas no cenário.

O Capítulo 5 mostra como se dá o estabelecimento dos cenários e respectivos escopos da ODS e da FS.

Hipótese-2:

Os artefatos que a ODS deverá produzir em uma iteração para serem enviados para a FS dependem do escopo de atuação da ODS e da(s) FS(s) envolvidas na iteração em foco.

Como comprovar a hipótese-2:

Os cenários que podem ser estabelecidos para uma iteração podem levar a duas situações:

- Cenários onde a ODS não possui um escopo operacional. Seu papel será unicamente o controle das interações com as FSs envolvidas no cenário através do processo a ser definido no presente trabalho.
- Cenários onde a ODS, além de fazer o controle das interações com as FSs envolvidas, ela também executará um escopo operacional produzindo artefatos para serem enviados para uma FS.

A apresentação dos escopos feita no Capítulo 5 mostra a referida interdependência deles para compor um cenário.

Hipótese-3:

Os artefatos necessários para que uma FS possa gerar o produto solicitado em uma iteração e o faça de forma desacoplada da ODS, depende do escopo da FS na iteração em foco, da correta utilização da UML na especificação desses artefatos e do pleno conhecimento em UML por parte dos profissionais da FS envolvidos no desenvolvimento dos produtos solicitados.

Como comprovar a hipótese 3:

A comprovação dessa hipótese requer a execução de um projeto de software com uso de pelo menos uma FS atuando nas iterações do projeto. Nesta ocasião a FS receberá da ODS um conjunto de artefatos (desenvolvido por ela ou por uma outra FS) especificados com uso da UML. Requer ainda que a FS possua pessoas com fluência em UML para ler e entender as especificações recebidas e elaborar o produto definido para o escopo. Para a comprovação dessa hipótese foi elaborada uma pesquisa-ação, a qual está descrita no Capítulo 6. A hipótese será comprovada se os artefatos produzidos pela FS forem aceitos pela equipe de qualidade da ODS e não tenha havido comunicação verbal entre a ODS e a FS utilizada.

Hipótese 4:

Para que a ODS possa controlar as interações com uma ou mais FSs envolvidas no cenário de cada iteração, é necessário um processo que execute o planejamento dos cenários e respectivos escopos da ODS e de cada FS envolvida na iteração e controle a execução das disciplinas operacionais do RUP dentro dos escopos da ODS e das FS envolvidas.

Como comprovar a hipótese 4:

Para a comprovação dessa hipótese se faz necessário testar o processo de controle das interações entre a ODS e FS conforme demonstrado na figura 12. Além do processo, também serão testados alguns mecanismos de controle, tais como a Solicitação de Serviços (SS) à FS, a Ordem de Serviço Interno (OSI), normas, padrões e *templates* que serão elaborados para ajudar no controle das interações entre essas duas organizações. A comprovação da hipótese será obtida através da pesquisa-ação detalhada no Capítulo 6, ocasião em que o processo e os demais mecanismos serão testados ao longo de várias iterações até que todos os mecanismos sofram os devidos refinamentos para que a equipe de qualidade da ODS não mais detecte falhas no produto final da iteração. Ou seja, a iteração precisará ser encerrada com nenhuma falha relativa à comunicação ODS/FS.

Para essa comprovação foi definido um cenário padrão para todas as iterações das quatro fases do RUP conforme demonstrado na figura 16 – Estratégia de Cenários e Escopos da Pesquisa-Ação. Todos os cenários serão idênticos para manter um comportamento padrão das organizações envolvidas ao longo do projeto. Cada cenário terá atuação da ODS e de duas FSs, cada uma com um escopo diferente. O Capítulo 6 irá demonstrar os detalhes da execução do processo pela ODS, tanto para executar seu escopo, quanto para controlar as interações ODS/FS. Se a equipe de qualidade homologar o produto final da iteração, a hipótese será considerada comprovada.

2.2 Projeto usado na pesquisa-ação

O projeto usado na pesquisa-ação pertence a uma empresa de consultoria em sistemas de informação. O dono do projeto é uma empresa do Rio de Janeiro, fato que permitiu deslocar para aquela cidade um Analista de Negócio e um Analista de Sistemas para executar, respectivamente, as disciplinas de Modelo do Negócio e Requisitos. Essas duas disciplinas compuseram o escopo que foi executado pela ODS. As demais disciplinas operacionais do RUP foram executadas por duas FSs.

Todas as interações do ciclo de vida do projeto utilizaram um único tipo de cenário o qual tiveram três escopos. Um executado pela ODS e dois executados respectivamente por duas diferentes FSs. O Capítulo 6 irá detalhar a execução dessa pesquisa-ação.

2.3 Ferramentas utilizadas

A pesquisa utilizou as seguintes ferramentas de software:

- Como ferramenta CASE (*Computer Aided Software Engineer*) foi escolhido o Enterprise Architecture (EA), ferramenta produzida pela SPARX (http://www.sparxsystems.com.au/).
- Modelagem de dados e projeto de banco de dados foi escolhida o ERwin da *Computer Associates* (http://www.ca.com.br/).
- A FS utilizou-se da plataforma Java, *Eclipse, Struts, Hibernate e Apache Tomcat.*
- Como sistema gerenciador de banco de dados (SGBD): Oracle10g.

Capítulo 3 – Processo de Software

3.1 Conceitos Introdutórios

O termo “processo” é muito utilizado no contexto de negócio e na Engenharia de Software. O significado semântico da palavra processo não se restringe aos processos estruturados. Em Engenharia de Software, qualquer forma de desenvolvimento constitui um processo, por mais imatura ou caótica que possa ser. O conceito de processo não especifica se ele é bom ou ruim, ou se leva ou não a um produto de qualidade [REI2001].

A presença de um processo, por mais simples que seja, permite que a equipe de projeto não se perca na seqüência de atividades que precisam ser executadas, evitando “zigue-zagues”, elevando a produtividade.

A literatura é farta em definições de Processo de Software. Seguem abaixo algumas dessas definições:

- Conjunto completo de atividades necessárias para a transformação de requisitos dos usuários em um sistema de software. Um processo é um *template* para a criação de projetos [JAC1999].
- Arcabouço de tarefas necessárias para a construção de software de alta qualidade [PRE2002].
- Uma seqüência de passos executados para alcançar um determinado propósito [IEEE 610.12-1990].

A OMG [OMG2005] definiu o *Software Process Engineering Metamodel* (SPEM) como um metamodelo que rege a definição de um Processo de Software para uma organização de desenvolvimento de sistemas. Esse metamodelo é

estruturado em quatro níveis de modelagem conforme mostrado na Figura 2, os quais serão detalhados na seção 3.2 do presente trabalho.

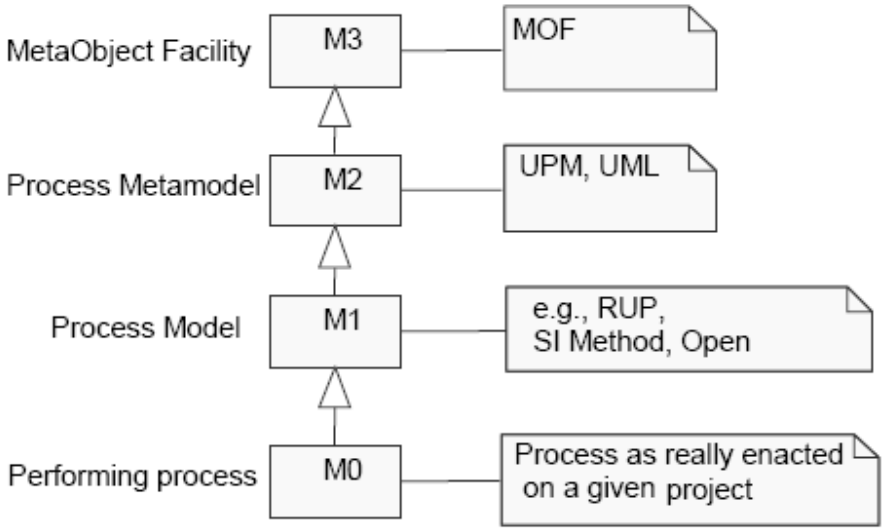


Figura 2 - Nível de modelagem do processo de software [OMG2005]

O nível M1 corresponde ao Modelo de Processo, em que um dos exemplos é o *Rational Unified Process* (RUP). O nível M0 representa o processo que é instanciado a partir do Modelo de Processo (pertencente ao nível M1) para ser efetivamente usado em um determinado projeto de desenvolvimento de software. Concluímos, portanto, que cada projeto deverá ter seu próprio processo de desenvolvimento, o qual é definido a partir de um modelo de processo.

Assim, projetos pequenos e simples não precisam aplicar o mesmo processo que deve ser seguido por um projeto complexo e de grande porte, que envolve uma grande equipe. Porém, ambos os processos podem ser derivados de um único Modelo de Processo. Cabe ao Gerente de Projeto definir um cenário (Processo de Desenvolvimento de Software), a partir do Modelo de Processo (ou Processo de Software) que será usado em um projeto específico.

A OMG [OMG2005] usa o termo Modelo de Processo como sinônimo de Processo de Software. Por outro lado, Pressman [PRE2002] usa o termo Modelo de Processo como sinônimo de Paradigma da Engenharia de Software, os quais serão detalhados na seção 2.2 do presente trabalho.

No contexto deste trabalho, o termo Processo de Software será usado como sinônimo de Modelo de Processo e o termo Processo de Desenvolvimento de Software como uma instância do Processo de Software pertencente ao nível M0 proposto pela OMG.

Quanto ao termo Metodologia de Desenvolvimento de Sistemas, vamos primeiramente, analisar a palavra Metodologia. Ciribelli [CIR2003] afirma que primeiramente devemos distinguir seus dois significados mais próximos: método e técnica: *"Entre método e técnica há uma diferença semântica análoga à que distingue o gênero da espécie. Método é o procedimento, ou o conjunto de procedimentos que servem para alcançar o fim da investigação. Técnica é o meio auxiliar que concorre para essa mesma finalidade. A diferença é que o método é geral e as técnicas são particulares". Ciribelli complementa ainda, afirmando que "Método é um procedimento geral baseado em princípios lógicos que podem ser comuns a várias ciências; Técnica é um meio específico usado em determinada ciência, é um aspecto específico desta ciência".*

Etimologicamente, Metodologia é uma palavra de origem grega composta pelos seguintes termos [CIR2003]:

- Meta = ao longo de.
- Odos = caminho.
- Logos = estudo, discurso.

Com base na etimologia e no conceito de Método acima descrito, podemos conceituar Metodologia, do ponto de vista da Engenharia de Software, como

uma apresentação de um caminho a ser seguido para se atingir um objetivo. Envolvendo o conceito de Técnica, também acima descrito, podemos afirmar que uma Metodologia pode utilizar uma ou mais Técnicas.

Juntando todos esses conceitos, podemos afirmar que Metodologia de Desenvolvimento de Software é um processo realizado com a utilização de técnicas bem definidas que levam à produção de um produto de software. Comparando com a definição de Processo de Desenvolvimento de Software definido no nível M0 proposto pela OMG [OMG2005], percebe-se, então, que Metodologia de Desenvolvimento de Sistemas pode ser utilizada como sinônimo de Processo de Desenvolvimento de Software.

3.2 Níveis de Modelagem do Processo de Software Proposto pela OMG

A *Object Management Group* (OMG) é uma organização internacional fundada em 1989, patrocinada por mais de seiscentos membros, incluindo fornecedores de sistemas de informação, desenvolvedores de softwares e usuários, cujo objetivo é promover teorias e práticas sobre metodologias Orientadas a Objetos para o desenvolvimento de *software* [OMG2002].

Para atender a esse objetivo, a OMG [OMG2005] desenvolveu um conjunto de abstrações definido como metamodelos que permite a definição de um processo de software orientado a objetos. Essas abstrações são distribuídas em níveis que compõem um *framework* arquitetural de processos chamado de Níveis de Arquitetura de Processo de Software, que podem ser usados para a criação de uma especificação de um processo de desenvolvimento de software, conforme mostrado na Figura 2.

3.2.1 Estabelecimento de Meta-objeto (*Meta-Object Facility MOF*)

É o nível de mais alta abstração (M3) da arquitetura proposta pela OMG (Figura 2), a qual corresponde ao Meta-Metamodelo representado pelo MOF (*Meta-Object Facility*), que define uma linguagem abstrata e um *framework* para especificar, construir e gerenciar metamodelos[8] independente de tecnologia. Define também um *framework* para implementação de repositórios que tratam metadados descritos através do metamodelos. Esses *frameworks* são usados pela indústria para a produção de repositórios de metadados com o objetivo de agilizar o processo de desenvolvimento [OMG2002].

Segundo [ZAN2005], o MOF pode ser usado para definir quaisquer tipos de metamodelos para domínios de interesse específicos. Pode-se citar como instâncias do MOF a *Unified Modeling Language* (UML), *Common Warehouse Metamodel* (CWM), XML *Metadata Interchange* (XMI) e o *Software Process Engineering Metamodel* (SPEM).

3.2.2 Metamodelo do Processo (*Process Metamodel)*

É no nível (M2) de abstração dentro da estrutura de processos de software sugerida pela OMG que trata de metamodelos usados para a criação de Modelos de Processos. Para esse nível da estrutura a OMG propõe o *Software Process Engineering Metamodel* (SPEM), o qual é uma derivação do MOF [OMG2005].

O SPEM é um metamodelo usado para a definição de modelo de processos e seus componentes. Não é usado para a customização de um processo de desenvolvimento de software pertencente ao nível M0 que é executado em um projeto de desenvolvimento. Seu uso está voltado para a definição de um modelo de processo que deverá pertencer ao nível M1. Por isso deve ser

[8] Metamodelo é um modelo que descreve outro modelo. Um modelo é uma instância de um metamodelo.

aplicado para o domínio da Engenharia de Software e não de Projeto de Software [OMG2005].

O SPEM fundamenta-se em três elementos básicos: papel, atividade e produto. O papel executa atividades as quais geram os produtos. Um produto, por sua vez, pode ser entrada para várias atividades. Um papel é responsável pelos produtos gerados pelas atividades. Um produto é sempre de responsabilidade de um único papel.

Do ponto de vista do processo, o SPEM baseia-se em uma estrutura chamada definição de trabalho, que é um elemento de elevado nível de abstração que pode se especializar em ciclo de vida, fase, iteração e atividade. Uma atividade pode ser composta por outras atividades, gerando produtos de trabalho os quais possuem uma tipificação.

Juntos, esses elementos descrevem as restrições e o comportamento do processo como um todo. Um ciclo de vida é uma definição de trabalho que possui uma seqüência de fases as quais podem ser subdivididas em iterações. Todos esses elementos possuem uma precondição e uma meta a ser alcançada.

3.3 Modelo de Processo de Software

Pressman [PRE2002] define o termo Modelo de Processo como um conjunto de elementos estratégicos usados no desenvolvimento de software, tais como o processo, métodos, ferramentas aplicadas ao longo do ciclo de vida de um software. Os modelos de processo de software propostos por Pressman são:

- Modelo seqüencial linear (*waterfall*).
- Modelo espiral.
- Prototipação.
- Incremental.

Neste capítulo, trataremos somente dois dos paradigmas acima, o Seqüencial Linear e o Espiral, porque tratam de conceitos que fundamentam o *Rational Unified Process*, descrito adiante, neste capítulo.

3.3.1 Modelo Seqüencial Linear

Também chamado de Processo em Cascata (*Waterfall*), é o mais popular. Originou-se na década de 1970 para dar suporte aos projetos de software para a plataforma alta de acordo com ROYCE citado por Soares [SOA2006].

Segundo Pressman [PRE2002], o Processo Seqüencial Linear está subdividido nas seguintes fases: Análise, Projeto, Codificação, Teste, Manutenção.

A fase de Análise consiste da definição de requisitos funcionais e não funcionais do software. O Projeto é a fase responsável pela definição da estrutura de dados, arquitetura do software, definição das interfaces e algoritmos para especificar o comportamento do software. A Codificação é responsável pela implementação do projeto em uma linguagem de computador. Teste é a fase em que o código produzido é testado para verificar se os requisitos foram atendidos conforme especificado na fase de Análise. Quando erros são identificados, o processo retorna para a fase de projeto ou de codificação para que os erros possam ser corrigidos. Após isso, novos testes são executados. Caso erros ainda persistam, o ciclo de correção é repetido até que nenhum erro seja identificado, depois do que o software é implantado no ambiente de produção. A fase de Manutenção ocorre quando o software já se encontra em produção e visa corrigir possíveis erros não percebidos por ocasião da fase de teste, ou adequar o software a mudanças no seu ambiente externo, ou ainda, adequação a mudanças funcionais ou no seu desempenho.

Vantagens:

- Simplicidade.

- Fácil controle.

Desvantagens:

- Nem sempre os projetos reais seguem a seqüência que o modelo propõe.
- Dificilmente todos os requisitos são identificados de uma única vez.
- A fase seguinte só pode ser iniciada após a conclusão da fase anterior.
- O cliente só tem contato com o software após a conclusão a fase de Codificação, a qual só inicia após a fase de Projeto. Isso demora e frequentemente gera desgastes da equipe de projeto.
- Eleva o risco do usuário não aprovar o software, fazendo com que todo o trabalho realizado seja prejudicado. Isso causa enorme perda de tempo, fato que eleva os custos do projeto.
- O tempo consumido entre a fase de Análise (em que os requisitos foram identificados) e o Teste (ocasião em que o usuário tem seu primeiro contato com o software em desenvolvimento), é grande dando tempo para que haja mudanças nos requisitos.

3.3.2 Modelo Espiral

O modelo de processo Espiral foi sugerido por Boehm [BOEHM1988] visando resolver uma série de dificuldades originárias do emprego do modelo de processo linear clássico (cascata).

O modelo espiral mostrado na figura 3 é formado por um conjunto de quatro ciclos de 360° onde cada ciclo corresponde a um nível de especificação ou fase do processo de desenvolvimento. Uma espiral (conjunto dos quatro ciclos) é usada para o desenvolvimento de um sistema completo, uma iteração, um componente ou qualquer subconjunto de um sistema. Para um projeto de desenvolvimento de um sistema particionado em dez iterações, seriam necessárias dez espirais para desenvolvê-lo.

Com exceção do último, cada ciclo de uma espiral está subdividido em quatro quadrantes. A junção dos ciclos, quadrantes e atividades formam um trinômio que define a arquitetura do modelo de processo espiral.

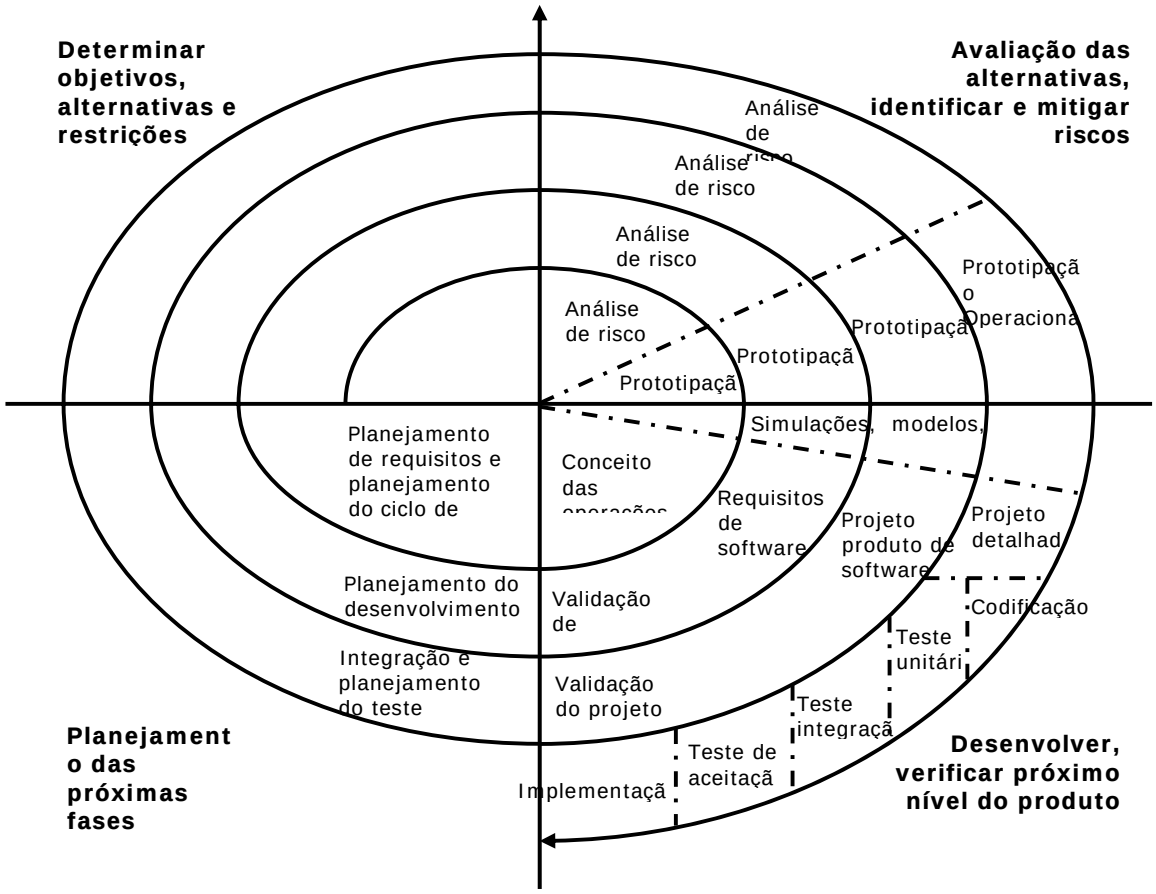


Figura 3 - Modelo de processo espiral de Boehm [BOE1988]

O primeiro quadrante reúne atividades responsáveis pela definição dos objetivos da porção do software que será desenvolvida. Trata ainda da definição de alternativas de soluções e as restrições impostas para o ciclo em questão da espiral. Essas atividades são comuns aos quatro ciclos da espiral.

O segundo quadrante faz uma avaliação das alternativas elencadas no primeiro quadrante. Trata principalmente de uma completa análise e mitigação dos riscos

para cada uma das alternativas identificadas. A principal ferramenta usada na mitigação dos riscos é a Prototipação. A análise de riscos e a prototipação são repetidas em todos os quatro ciclos.

O terceiro quadrante é o responsável pela realização das tarefas do desenvolvimento e a validação dos artefatos produzidos. Cada ciclo, dentro deste quadrante da espiral, faz um refinamento adicional na especificação do produto em desenvolvimento.

- No primeiro ciclo, é feito um modelo conceitual da operação.
- No segundo ciclo, é a ocasião em que é feita a elicitação e validação dos requisitos.
- O terceiro trata das atividades relacionadas ao projeto e a sua validação.
- O quarto (e último) ciclo utiliza-se de uma parte do modelo de processo Linear (cascata): projeto detalhado, codificação, teste unitário, integração, teste integrado, teste de aceitação e implantação em produção.

O quarto (último) quadrante trata do planejamento das próximas fases.

- No primeiro ciclo são feitos os planos de requisitos e do ciclo de vida.
- No segundo ciclo é feito o planejamento do desenvolvimento e
- No terceiro ciclo é feito o plano da integração e dos testes do produto gerado pelo espiral.

Em resumo, tem-se que o primeiro ciclo possui características de modelagem de mais elevado nível de abstração com uma abrangência de todo o sistema foco da espiral. O segundo ciclo tem um foco voltado para a gestão de requisitos. O terceiro ciclo trata do projeto do software em questão. O quarto ciclo responsabiliza-se pela construção do software para posterior implantação em um ambiente de produção.

3.4 Processo Unificado da Rational (Rational Unified Process - RUP)

O RUP é um produto de processo desenvolvido e mantido pela Rational Software (hoje de propriedade da IBM) que pode ser adaptado e estendido para compor as necessidades de uma organização que o esteja adotando. Foi desenvolvido para ser um *framework* de processos que podem ser adaptados e estendidos objetivando atender às necessidades de uma organização [KRU2001].

Conforma já mencionado neste capítulo, o RUP se enquadra na categoria de um modelo de processo elaborado a partir do SPEM. Por isso, não deve ser usado como uma instância a ser aplicada diretamente em um projeto, mas como um modelo a partir do qual se deve elaborar o processo a ser seguido, em função das idiossincrasias do projeto [OMG2005].

Seguindo o RUP, um sistema pode ter seu desenvolvimento segmentado em mais de um ciclo, onde cada um é formado por uma seqüência de quatro fases (Iniciação, Elaboração, Construção e Transição). Essas fases serão detalhadas mais à frente, neste capítulo. Cada fase do ciclo em foco é executada de forma iterativa. Cada iteração, por sua vez, é formada por um conjunto de disciplinas as quais contêm atividade que produzem artefatos.

3.4.1 Características do RUP

As principais características do RUP são:

- Incremental.
- Iterativo.
- Dirigido por casos de uso.
- Centrado em arquitetura.
- Antecipa a análise e mitigação dos riscos.
- Permite a reutilização de componentes.

Processo incremental

Essa característica permite ao gerente uma total customização do processo a ser seguido no projeto para melhor se adequar ao porte e complexidade do sistema. O processo é incremental por causa da sua característica cíclica, ou seja, entrega uma porção do sistema a cada ciclo.

O RUP utiliza o processo espiralado conforme proposto por Boehm [BOEHM1988], que permite que sistemas muito grandes e complexos possam ser “fatiados”, ou seja, permite que o sistema possa ser desenvolvido de forma incremental para facilitar o desenvolvimento e para antecipar a entrega de versões aos usuários.

Esta característica cíclica também é prevista no metamodelo SPEM [OMG2005] conforme descrito na seção 3.2.2 Metamodelo do Processo. A figura 4 é uma adaptação do modelo de processo espiral de Boehm [BOE1988] que mostra uma abstração dessa característica cíclica do RUP, onde cada quadrante corresponde a uma fase do RUP. Cada ciclo do processo passa pelas mesmas fases de Iniciação, Elaboração, Construção e Transição.

Não há paralelismo de execução de fases, ou seja, as fases são executadas sempre seqüencialmente. Cada ciclo produz uma versão do sistema, onde cada versão é um incremento da versão anterior. Um incremento (ciclo) gera um produto a ser entregue e utilizável pelos usuários. Cada incremento produz uma determinada quantidade de funcionalidades do sistema, de tal forma que após o último incremento teremos todo o sistema concluído.

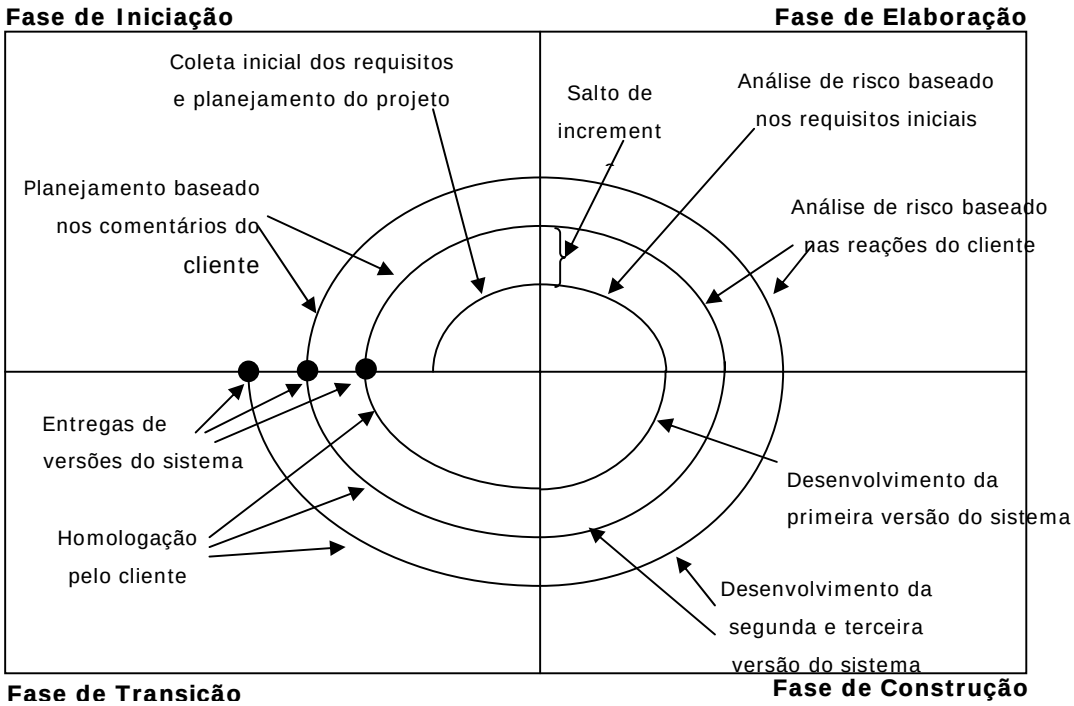


Figura 4 – Modelo de processo cíclico – incremental (adaptação do modelo espiral de Boehm [BOE1988])

Kruchten [KRU2001] sugere que o primeiro ciclo de vida deve ser chamado de Ciclo de Desenvolvimento e os demais ciclos devem ser chamados de Ciclos Evolutivos. A versão inicial é criada pelo Ciclo de Desenvolvimento. Essa versão entra em produção e passa ser usada normalmente pelos seus usuários. As versões seguintes são obtidas através da adição de novas funcionalidades desenvolvidas pelos Ciclos Evolutivos. Cada Ciclo Evolutivo entrega uma nova versão do sistema com a adição de novas funcionalidades.

Um projeto que usa o RUP requer que o planejamento dos incrementos produzidos pelos ciclos de desenvolvimento e evolutivo seja feito de tal modo que permita que cada versão possua interdependência das funcionalidades das versões futuras, para que já possa ser utilizada por seus usuários, na medida em que for sendo entregue.

Processo Iterativo

Uma iteração é uma característica também prevista pelo SPEM [OMG2005] (veja seção 3.2.2 Metamodelo do Processo). Segundo o RUP [RUP2002], uma iteração é uma seqüência distinta de atividades devidamente planejada, com critérios de avaliação, que resulta em um release (interno ou externo) do sistema. Uma Fase pode ter uma ou mais iterações. Cada iteração produz uma porção do sistema que é refinado nas iterações seguintes.

A execução de uma iteração causa um incremento no produto do projeto de desenvolvimento, porém menor que o incremento dado por um ciclo de desenvolvimento ou evolutivo. É um incremento dentro de um ciclo, ou seja, é um incremento dentro de outro incremento. O incremento causado por um ciclo é formado pela soma dos incrementos causados pelas suas iterações.

Cada iteração produz uma série de elementos formando um bloco[9] que se junta a outros blocos construídos em outras iterações, compondo incrementalmente um sistema. Um bloco é uma versão operacional de parte de um sistema que demonstra um subconjunto da capacidade do sistema [JAC1999].

Uma iteração é como um miniprojeto quase completo, no qual um fluxo de atividades é executado resultando, na maioria dos casos, num sistema executável, porém incompleto. Uma iteração inicia com o planejamento e requisitos e termina com um *release*, interno ou externo [KRU2001].

As atividades executadas em uma iteração pertencem às disciplinas. O RUP propõe nove disciplinas, as quais são: Modelagem do Negócio, Gestão de Requisitos, Projeto, Codificação, Teste, Implantação, Gestão de Projeto, Gestão de Configuração e Mudança e Gestão do Ambiente [KRU2001]. Essas disciplinas serão detalhadas ainda neste capítulo.

---

[9] Um bloco (em inglês *build*) é uma versão operacional do sistema que demonstra uma parte da capacidade do sistema [JAC1999; pg.91].

A Figura 1 mostra o que é popularmente chamado de “gráfico de baleias”, em que, no lado esquerdo, dispõem-se as disciplinas, cujas atividades podem ser executadas nas iterações. Acima, estão dispostas as fases e, abaixo, as referências das iterações numeradas dentro das respectivas fases.

Na página inicial da sua “Visão Geral”, o RUP [RUP2002] mostra um outro aspecto importante do “gráfico de baleias” que é a sua bidimensionalidade, ou seja, possui uma característica cartesiana onde no eixo do “X” temos o tempo que é consumido por ocasião da execução das fases. No eixo do “Y”, temos a disposição das disciplinas. A interseção dessas duas dimensões representa as atividades realizadas pelo fluxo das iterações.

Uma cuidadosa observação do “gráfico de baleias” mostra, através das corcundas, os graus de intensidade em que as atividades das disciplinas são executas nas diferentes fases são diferentes. Podemos notar que as disciplinas de Gestão de Requisitos e Análise-*Design* são usadas praticamente em todas as fases. Contudo, pode-se notar que a Gestão de Requisitos é iniciada antes da Análise-*Design*. Comparando a disciplina de Análise-*Design* com a Implementação, percebe-se que a Implementação é usada nas duas iterações da fase de Transferência, enquanto que a Análise-*Design* termina na primeira iteração da fase de Transferência. A disciplina de Modelagem de Negócio inicia-se intensamente na fase de Iniciação e vai diminuindo a intensidade de uso ao longo do tempo para quase não mais haver na fase de Transição.

Obviamente, esse gráfico é uma abstração e não se espera que nos projetos de desenvolvimento ocorra tudo exatamente como está representado no gráfico. Essa abstração objetiva orientar o gerente do projeto na elaboração do *Work Breakdown Structure* (WBS) e no planejamento das iterações, bem como todo o acompanhamento do projeto.

Figura 5 - Plano de iteração da fase de Iniciação [RUP2002]

O RUP [RUP2002] sugere que as atividades das disciplinas podem ser executadas paralelamente intra e interdisciplinas. O RUP oferece quatro exemplos de planejamento de iterações, uma para cada fase. A Figura 5 é um exemplo de planejamento para a fase de Iniciação. As fases são executas seqüencialmente, porém as atividades dentro de uma iteração, independentemente da fase, podem ser executadas paralelamente.

Processo dirigido por casos de uso

O RUP é uma abordagem dirigida por caso de uso. Isso significa que o caso de uso definido para um sistema é a base para o processo de desenvolvimento inteiro [KRU2001].

Um caso de uso é uma coleção de cenários que descrevem um procedimento interativo entre o sistema e um ator, produzindo um resultado de valor do ponto de vista do ator. Um ator pode ser uma pessoa, um outro sistema, um componente do sistema operacional ou um dispositivo eletrônico que interage diretamente com o sistema. Um caso de uso não descreve o comportamento interno do sistema. Essa descrição é feita nas atividades da disciplina Análise-*Design* da iteração [BOO1999].

A grande maioria dos casos de uso é identificada nas fases de Iniciação e Elaboração dentro da disciplina Gestão de Requisitos conforme demonstrado no “gráfico de baleias” (Figura 1). É na disciplina de Análise-*Design*, que um caso de uso é totalmente “dissecado” para verificar como ele é realizado internamente pelo sistema. Este procedimento se repete em todas as iterações, principalmente nas fases de Elaboração e Construção, conforme demonstrado no gráfico de baleias.

Todo o planejamento de iterações é feito a partir do modelo de casos de uso. É a partir do Modelo de Casos de Uso que são elaborados o Modelo de Análise,

Modelo de Design, Modelo de Implementação, Modelo de Implantação e Modelo de Testes. É por isso que é dito que o RUP é um processo dirigido por casos de uso [JAC1999].

3.4.2 Fases de um Ciclo de Desenvolvimento

A característica seqüencial da execução das fases do RUP oferece uma dimensão relacionada ao tempo na sua abstração bidimensional. O RUP possui quatro fases: Iniciação, Elaboração, Construção e Transferência. Essas quatro fases serão descritas nas sessões a seguir.

Fase de Iniciação

Segundo Jacobson [JAC1999], o foco principal da fase de Iniciação é estabelecer o caso de negócio que será levado avante pelo projeto. Esta fase não faz um estudo completo de todo o sistema, mas é a ocasião em que os casos de uso necessários para a sua iniciação são identificados para que se possa executar as seguintes atividades:

- Definir o escopo do sistema a ser desenvolvido.
- Iniciar a identificação das interfaces com outros sistemas com os quais interage.
- Elaborar a Descrição da Arquitetura candidata para o sistema, focando especialmente os aspectos mais desconhecidos, riscos ou qualquer outra dificuldade. Somente em situações especiais e bastante raras é que, na fase de Iniciação, um protótipo executável é construído.
- Identificar os riscos críticos, aqueles que afetam o sucesso do projeto de desenvolvimento do sistema. Uma análise dos riscos é feita com o objetivo de eliminá-los ou de mitigá-los. Riscos não críticos podem ser postergados para a fase de Elaboração.
- Fazer uma análise dos problemas dos usuários e patrocinadores do sistema, juntamente com suas necessidades, para direcionar o sistema

em busca da solução desses problemas e atendimento das necessidades declaradas.

Kruchten [KRU2001] sugere que a fase de Iniciação deverá obter o consentimento dos stakeholders nos objetivos do ciclo de vida do projeto. Sugere ainda que os produtos finais da fase são:

- Documento de Visão contendo uma visão geral dos principais requisitos do projeto, características e as principais restrições.
- Modelo de casos de uso (com os já identificados até o momento).
- Glossário contendo os termos já identificados. Opcionalmente pode-se desenvolver um Modelo de Domínio mais completo que um Glossário.
- O Caso de Negócio fornecendo as informações necessárias do ponto de vista de um negócio:
    - Para determinar se vale ou não a pena investir no projeto. Se o produto de software tiver objetivos comerciais, o Caso de Negócio deve incluir um conjunto de suposições sobre o projeto e a ordem de importância do Retorno do Investimento (ROI). Essas suposições são verificadas novamente no fim da fase de Elaboração, quando o escopo e o plano estiverem definidos com mais exatidão [RUP].
- Uma avaliação inicial dos riscos.
- Um plano do projeto mostrando as fases e as iterações. T676

Fase de Elaboração

A meta da fase de elaboração é criar a baseline para a arquitetura do sistema a fim de fornecer uma base estável para o esforço da fase de Construção.

Segundo Jacobson [JAC1999], *“a arquitetura de software é um conjunto de decisões significantes sobre a organização de um software, tais como:*

- *Seleção dos elementos estruturais e interfaces com os quais o software é composto, junto com seu comportamento como especificado nas colaborações entre esses elementos;*
- *Composição desses elementos estruturais e comportamentais em subsistemas progressivamente maiores;*
- *Estilo arquitetural que guia essa organização, tais como os elementos e suas interfaces, suas colaborações e suas composições.*

Arquitetura de software é consistida não só por estrutura e comportamento, mas com uso, funcionalidade, performance, flexibilidade, reuso, compreensibilidade, restrições econômicas e tecnológicas, intercâmbio e preocupações estéticas".

As iterações da fase de Elaboração também são planejadas a partir dos casos de uso. Dentre os casos de uso existentes, são escolhidos aqueles estratégicos do ponto de vista da arquitetura, ou seja, os casos de uso relacionados aos riscos de não se obter uma arquitetura estável. Esses casos de uso são aqueles que cobrem as tarefas ou funções que o sistema irá realizar. Para se achar quais são esses casos de uso, pode-se fazer a pergunta: "Para que estamos fazendo esse sistema?" A resposta a essa pergunta indicará quais os casos de uso estratégicos do ponto de vista arquitetural. Adicionalmente a esses, relacionam-se os casos de uso associados aos requisitos não-funcionais, tais como: desempenho, segurança, interoperabilidade, etc. [JAC1999].

Para testar a arquitetura, freqüentemente é construído um protótipo arquitetural visando a eliminar ou mitigar os riscos envolvidos. O protótipo arquitetural também envolve o desenvolvimento de interfaces gráficas para a validação pelos usuários. Exaustivos testes são realizados para verificar se o protótipo atende aos requisitos e, quando necessário, o projeto e o protótipo arquitetônico sofrem modificações em busca da solução ideal.

Com os riscos identificados, analisados e eliminados ou mitigados, tem-se uma arquitetura consistente, testada e homologada pelos envolvidos, podendo-se, dessa forma, dar início à fase de Construção.

Fase de Construção

O objetivo da fase de construção é concluir o detalhamento dos casos de uso que ainda não foram tratados nas fases de iniciação e elaboração. A partir desses casos de uso, é feito um plano de iterações no qual grupos de casos de uso são desenvolvidos por meio das atividades das disciplinas de analise, projeto, construção, testes e implantação, seguindo a baseline da arquitetura definida por ocasião da fase de Elaboração. No final da última iteração da fase de Construção, tem-se um software pronto para liberação de uma versão inicial, comumente chamada de “beta-release”. De certa forma, a fase de construção é um processo de manufatura, em que a ênfase está no gerenciamento de recursos e no controle das operações visando a otimizar custos, programações e qualidade [RUP2002].

Fase de Transição

É a última fase do projeto. É o último quadrante de um ciclo de desenvolvimento descrito na Figura 4. Nesta fase, todas as atividades estão voltadas para que o software esteja disponível para seus usuários. Pode possuir várias iterações votadas principalmente para as disciplinas de teste e implantação. Devido ao contato maior dos usuários com o sistema, freqüentes ajustes podem ocorrer para o refinamento do sistema visando a melhor atender às necessidades dos usuários [RUP2002].

A Gestão de Configuração tem uma participação extremamente ativa e coordena todas as demais atividades, em busca do empacotamento de todos os artefatos em uma baseline da versão final do ciclo do projeto.

3.4.3 Disciplinas

O RUP [RUP2002] apresenta a seguinte definição de disciplina: “é um conjunto de atividades relacionadas a uma área de interesse importante em todo o projeto. O principal objetivo do agrupamento de atividades em disciplinas é ajudar a compreender o projeto a partir de uma perspectiva de um processo em cascata (tradicional)”.

Conforme já mencionado anteriormente, existem duas categorias de disciplinas: Operacionais e Gerenciais. As disciplinas operacionais são aquelas que executam atividades de caráter operacional no processo produtivo de artefatos intermediários[10] e artefatos de software. As disciplinas gerenciais executam atividades que planejam e controlam o processo produtivo. Em resumo, no presente trabalho, as disciplinas estão categorizadas da seguinte maneira:

- Disciplinas Operacionais:
    - Disciplinas de Produção.
        - Disciplinas de produção de artefatos intermediários: Modelagem do Negócio, Requisitos e Análise-*Design*.
        - Disciplina de produção de artefato de software: Implementação.
    - Disciplinas de Suporte à Produção.
        - Teste.
        - Implantação.
- Disciplinas gerenciais:
    - Gestão de Configuração e Mudança.
    - Gestão de Projetos.
    - Ambiente.

[10] Artefato intermediário é um termo usado como sinônimo do termo “Produto Intermediário” introduzido pela [ISO9126], o qual é um artefato elaborado por uma atividade prevista no processo de software como um meio para a produção final de um produto de software. Veja maiores detalhes na seção 6.5.1 (Proesso de Qualidade).

Modelagem do Negócio

Os objetivos da Modelagem do Negócio são: 1) Entender a estrutura e a dinâmica da organização alvo do sistema. 2) Entender os problemas da organização e identificar possíveis melhoramentos. 3) Garantir que o patrocinador, usuário e os desenvolvedores tenham o mesmo entendimento da organização. 4) Derivar os requisitos do sistema necessários para suportar a organização alvo do sistema [KRU2001]. A Modelagem de Negócio é uma ferramenta que promove o alinhamento do sistema em desenvolvimento com o negócio em foco.

A Modelagem de Negócio nem sempre deve ser usado no processo de desenvolvimento. Sua utilidade é maximizada quando o ambiente do negócio em que o sistema está inserido é complexo e quando o número de atores é elevado [LEF2000].

Os principais produtos da Modelagem de Negócio são: Modelo de Casos de Uso de Negócio e Modelo de Objetos de Negócio. É a partir desses artefatos que se deriva o Modelo de Casos de Uso e o Modelo *de Design*.

Gestão de Requisitos

Gestão de requisitos é uma abordagem sistemática para elicitar, organizar e documentar os requisitos de um sistema. Também trata do processo que estabelece e mantém o acordo entre o cliente e a equipe do projeto na mudança dos requisitos do sistema [LEF2000].

Os produtos da Gestão de Requisitos são o Documento de Visão, e a Especificação de Requisitos de Software (ERS), a qual é constituída pelo Modelo de Casos de Uso e pelo Documento de Requisitos [LEF2000]. O Modelo de *Design*, Modelo de Teste e os Manuais do Usuário e o material de treinamento são desenvolvidos a partir dos produtos da Gestão de Requisitos.

Análise-*Design*

No Processo Unificado [JAC1999] a Análise e o Projeto são disciplinas separadas, mas no RUP [RUP2002], elas são reunidas em uma única disciplina. No presente trabalho usaremos a abordagem do RUP. Os objetivos dessa disciplina são: 1) Transformar os requisitos em um projeto do sistema. 2) Desenvolver a arquitetura do sistema. 3) Desenvolver uma solução de implementação visando a obtenção de um desempenho esperado.

Analisando o gráfico de baleias (Figura 1), percebe-se que a maior intensidade de Análise-*Design* acontece na fase de Elaboração. Essa elevada intensidade se deve ao fato de que é na fase de Elaboração que a Arquitetura é descrita. No início da fase de Construção a Análise-*Design* ainda é intensa (mas menos que na fase de Elaboração), mas vai decaindo nas iterações seguintes. Isso ocorre porque na medida que as iterações vão se concluindo, menos funcionalidades restam para ser analisadas. Geralmente, nas iterações finais da Construção ocorre desenvolvimento de trechos do projeto que já tiveram algum tratamento nas iterações anteriores.

O principal artefato produzido é o Modelo de *Design*, o qual é um modelo de objeto que descreve a realização de casos de uso e serve como uma abstração do modelo de implementação e seu código-fonte [RUP2002].

As características de um modelo de *design* são:

- Realiza os casos de uso e os requisitos não-funcionais.
- Elevada manutenibilidade, resistindo a mudanças nos casos de uso, requisitos não-funcionais e no ambiente de implementação.
- Oferece um claro suporte na modelagem da implementação.
- Não possui informações relacionadas à codificação, tais como: dicas, sugestões, trechos de código, etc.
- Adapta-se facilmente a mudanças efetuadas em requisitos.

Implementação

Como o próprio nome diz, é a disciplina que trata das atividades relacionadas à codificação. Segundo RUP [RUP2002], essas atividades têm o objetivo de:

- Definir a organização do código em termos de subsistemas de implementação organizados em camadas.
- Implementar classes e objetos em termos de componentes (arquivos-fonte, binários, executáveis e outros).
- Testar unitariamente os componentes desenvolvidos.
- Integrar os resultados produzidos por implementadores individuais (ou equipes) ao sistema executável.

É durante a implementação que os blocos são desenvolvidos. Um bloco (em inglês *build*) é uma versão operacional do sistema que demonstra uma parte da capacidade do sistema [JAC1999]. Um bloco sofre incrementos por ocasião de uma iteração. Cada iteração produz novas capacidades que são integradas à versão anterior do bloco criando uma nova versão. Cada bloco é colocado sob o controle da configuração nos casos em que haja necessidade de voltar a uma versão mais antiga, quando a funcionalidade adicionada causar quebras ou qualquer outro efeito que comprometa a integridade do *build* [RUP2002].

Testes

É a disciplina que garante a qualidade do produto de software. Essa garantia é obtida da seguinte forma:

- Localização e documentação dos defeitos.
- Verifica se os requisitos foram atendidos e se sua implementação foi feita de forma adequada.
- Verifica se o projeto foi seguido na implementação.

Uma característica importante das atividades dessa disciplina é que, enquanto as disciplinas de Gestão de Requisitos, Análise-*Design* e Implementação buscam abrangência, os Testes buscam deficiência [RUP2002].

Teste de software não é uma fase do projeto nem uma atividade realizada uma única vez. É uma disciplina que se faz presente ao longo do ciclo de vida de um projeto, em cada iteração de cada uma das quatro fases do projeto. Em função disso, Kruchten [KRU2001] propõe três dimensões de teste de software:

- Dimensão da qualidade: define quais características de qualidade de software deverão ser testadas. A norma ISO 9126 [ISO9126] sugere um conjunto de seis características de qualidade de software: funcionalidade, confiabilidade, usabilidade, eficiência, manutenibilidade e portabilidade.
- Estágio do teste: define o momento, dentro do ciclo de vida do projeto, que o teste será executado. Cada estágio abrange um escopo crescente. O Teste Unitário abrange um elemento do sistema e pode ser executado logo após sua implementação. O Teste de Integração tem um escopo mais abrangente, pois reúne um conjunto desses elementos, já testados unitariamente, para verificar seu comportamento em conjunto com os demais elementos. O Teste de Sistema tem o escopo de toda a aplicação, reunindo todos os elementos já testados nos estágios unitários e integração, adicionando-se ainda, todos os sistemas com os quais o sistema deverá interagir. Este teste simula o ambiente de produção onde o sistema deverá ser instalado. Até este instante, estes testes são feitos sem a participação dos usuários. O próximo estágio é o Teste de Aceitação, o qual envolve a participação dos usuários finais.
- Tipo de teste: define uma abordagem usada em um enésimo teste aplicado sobre uma característica de qualidade e em um determinado estágio de teste. Kruchten [KRU2001] sugere os seguintes tipos de teste: teste padrão, teste de configuração, teste de função, teste de instalação,

teste de integridade, teste de carregamento, teste de desempenho e teste de tensão.

Os testes de software fazem parte de um processo maior chamado de Processo de Qualidade. A evolução dos eventos dentro do Processo de Qualidade precisa ser documentada. A norma IEEE [IEEE 829-1998] define os tipos de documentos que podem ser desenvolvidos ao longo do processo. Esses documentos são: Plano de Teste, Especificação do Projeto de Teste, Especificação de Teste Case, Especificação dos Procedimentos de Teste, Relatório de Transmissão de Item de Teste, Log de Teste, Relatório de Incidente de Teste, Relatório do Resumo do Teste.

Implantação

Dentro do fluxo operacional de uma iteração, a Implantação é a última etapa. Suas atividades são executadas com mais freqüência no fim da fase de Elaboração e intensamente na fase de Transferência, que é a fase que o sistema é disponibilizado para uso pelos seus usuários. As principais atividades são [KRU2001]:

- Teste no ambiente operacional (beta-teste).
- Empacotar o sistema para entrega.
- Distribuir e instalar o sistema.
- Treinar os usuário finais.
- Migrar o sistema existe ou converter o banco de dados.

Disciplinas gerenciais do RUP

As disciplinas gerenciais do RUP são:

- Gestão de projetos.
- Gestão de configuração e mudança.
- Ambiente.

Essas disciplinas não serão detalhadas neste trabalho porque suas execuções independem das variações entre os diversos tipos de cenários.

Capítulo 4 – Fábrica de Software

O termo “Fábrica de Software” surgiu em meados da década de 80, mas somente na década de 90 é que o termo foi usado no Brasil por empresas prestadoras de serviço em tecnologia da informação.

Segundo Aaen [AAE1997], o termo “fábrica” leva a uma comparação com a produção em massa de produtos industriais, mas isso é plenamente discutível. Isso pode facilmente levar a uma ilusão com respeito aos tipos de intervenções que podem ser feitos na fábrica de software que, de fato, podem melhorar as operações de software. Contudo, para muitas pessoas o conceito de fábrica também implica uma forma particular de organização do trabalho, com um considerável nível de especialização, formalização de comportamento e padronização dos procedimentos de trabalho.

Costa [COS2003] apresenta um estudo sobre as fábricas de software brasileiras e afirma o seguinte:

- Geralmente, atuam apoiando-se na proposta de estruturação de uma célula de programação externa que funciona de forma complementar à equipe de informática do cliente, com o objetivo de atender a uma demanda média de programação e os eventuais picos que venham a surgir durante o processo do cliente.
- Utiliza-se de suas próprias instalações, ou as do cliente, agrupando recursos humanos e tecnológicos em células especializadas para atender especificamente a um cliente, executando todo o ciclo de vida de um sistema.

Essa especificidade no atendimento a um cliente causa um elevado acoplamento entre a ODS e a FS. Para solucionar esse problema, Costa [COS2003] propõe um modelo de Fábrica de Software Padrão que possui um processo de recepção de solicitação de serviços de clientes, composto por um conjunto de atividades, padrões, métricas e técnicas consagradas pela Engenharia de Software, adaptadas para a cultura nacional e que devem ser aplicados na avaliação das solicitações de serviços recebidos das ODS clientes.

A utilização do termo “Fábrica de Software” requer alguns cuidados, pois não é qualquer organização de desenvolvimento de software que pode ser classificada com uma fábrica de software. Segundo Fernandes [FER2004], a expressão “Fábrica de Software” é usada com várias conotações, mas a maioria das empresas de serviço usa a expressão para se referir a uma “Fábrica de Programas”.

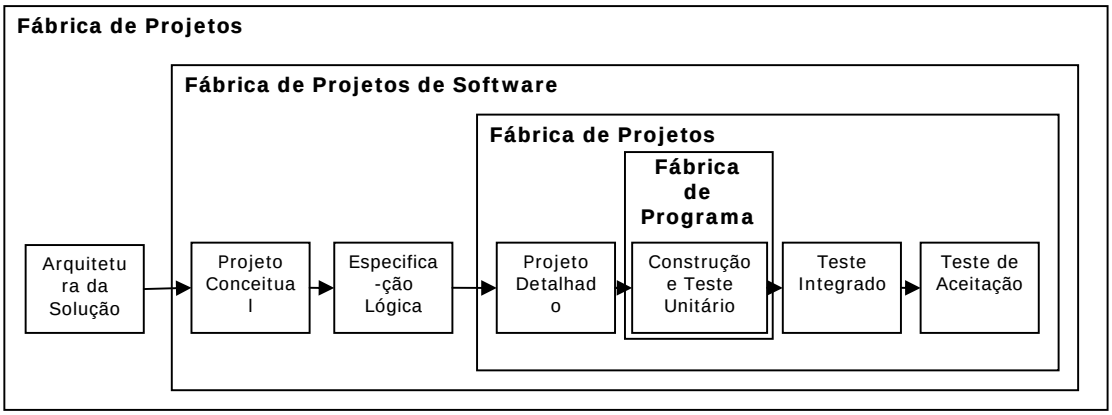


Figura 6 - Possíveis escopos de atuação de uma fábrica de software [FER2004]

Fernandes [FER2004] definiu vinte e duas características de uma fábrica de software, mostrando que não é qualquer pool de programadores que vem a constituir uma fábrica de software.

Uma fábrica pode atuar em diferentes escopos, conforme demonstrado na Figura 6. Os escopos são definidos a partir de um processo em cascata. A Arquitetura da Solução é uma fase que define uma visão sistêmica da empresa, levando em consideração os processos de negócio, o modelo de domínio referente aos processos e um mapeamento para os sistemas de informação. Isso mostra as interdependências entre os sistemas. A Arquitetura da Solução também mostra as tecnologias utilizadas pela empresa para serem usadas pelos sistemas a serem desenvolvidos.

O Projeto Conceitual equivale ao modelo de casos de uso de todo o sistema. No RUP, esse modelo obtido incrementalmente por meio das iterações desenvolvidas ao longo das quatro fases.

A Especificação Lógica define a lógica do sistema, porém sem um compromisso com uma solução de tecnologia a ser aplicada. Seu objetivo é definir a lógica do comportamento do sistema. No RUP, equivale ao Modelo de Análise, que é um dos produtos da disciplina de Análise-*Design*, cujas atividades são executadas em cada iteração de cada fase.

Projeto Detalhado define o sistema em função da plataforma tecnológica a ser utilizada. Equivale ao Modelo de *Design*, Modelo de Implementação e Modelo de Implantação do sistema, os quais no RUP, são obtidos na disciplina de Análise-*Design* executada nas iterações de cada fase.

Construção e Teste Unitário correspondem à disciplina de implementação do RUP. Os testes Integrados e de Aceitação, correspondem à disciplina de Teste do RUP.

Conclui-se, assim, que os escopos definidos por Fernandes [FER2004] para a fábrica de software toma como princípio um processo linear clássico.

Uma observação cuidadosa da Figura 6 mostra que, com exceção do escopo Fábrica de Programas, todos os demais escopos envolvem a fase de Teste Integrado e Teste de Aceitação. Isso sugere que esta modalidade de fábrica (Fábrica de Programas) recebe uma especificação, codifica, faz seus testes internos na fábrica e entrega ao cliente contratante, o qual ficará responsável pela integração do sistema como um todo e elaborar os testes integrados e de aceitação.

O presente trabalho toma como definição de fábrica de software o seguinte:

> Fábrica de Software é uma organização que provê serviços de desenvolvimento de sistemas com as seguintes características:
>
> - Atuação em vários cenários de desenvolvimento, onde cada cenário possui um escopo de atuação, com um processo bem definido e um conjunto de padrões a serem fielmente seguidos para o atendimento de solicitações de serviços de clientes.
> - Detentora de mão de obra altamente especializada alocada em unidades organizacionais igualmente especializadas.
> - Constante busca de reutilização de padrões, modelos, componentes e experiências obtidas em projetos anteriores.
> - Utilização de ferramentas adequadas para prover elevada produtividade.
> - Possui mecanismos que proporcione contínuo melhoramento dos processos em busca de melhor produtividade, qualidade e redução de custos.
> - Possui um processo de recepção de solicitação de serviços que valida os insumos enviados pelo cliente para a produção do software com a qualidade esperada.
> - O processo produtivo deverá ser desacoplado do cliente, ou seja, todos os insumos validados na recepção da solicitação de serviços deverão ser suficientes para a produção do software esperado.

Alguns estudos [ROC2004], [BRI2004], [MRQ2004-a], [MRQ2004-b] [CAB2006] foram feitos pelo Centro de Informática da Universidade Federal de Pernambuco com FS usando o RUP como modelo de processo. Estes estudos abordam a utilização do RUP preparando a FS para atuar em todo o ciclo de vida do projeto. Com isso, o controle do processo está na FS, e a ODS não tem nenhuma participação operacional no processo de desenvolvimento, ou seja, ela não executa nenhuma das disciplinas operacionais do RUP.

Capítulo 5 – Mecanismos de controle da interação entre a ODS e FS

5.1 Introdução

O objetivo deste capítulo é definir os mecanismos de controle entre a ODS e uma ou mais FSs para a produção de um sistema desenvolvido de forma iterativa, usando o RUP como modelo de processo de software e a UML como linguagem de notação. Esses mecanismos estão associados às hipóteses 1 e 2 as quais estão descritas no Capítulo 1.

Hipótese-1:

Os pontos dentro do fluxo de uma iteração nos quais a ODS deverá acionar a FS dependem do cenário estabelecido para a iteração em foco, onde cada uma dessas organizações deverá ter um escopo de atuação executando atividades pertencentes às disciplinas operacionais do RUP.

Para comprovar essa hipótese, faz-se necessário o estabelecimento de:

- todas as possibilidades de escopos, em que cada um é constituído por uma ou por mais disciplinas do RUP;
- todas as possibilidades de cenários, em que cada um é um arranjo de escopos executados pela ODS e pelas FSs envolvidas no cenário.

Hipótese-2:

Os artefatos que a ODS deverá produzir em uma iteração para serem enviados para a FS dependem do escopo de atuação da ODS e da(s) FS(s) envolvidas na iteração em foco.

Neste capítulo, serão mostrados cenários em que a ODS pode atuar de duas formas:

- Cenários onde a ODS não possui um escopo operacional. Seu papel será unicamente o controle das interações com as FSs envolvidas no cenário por meio do processo a ser definido no presente trabalho.
- Cenários em que a ODS, além de fazer o controle das interações com as FSs envolvidas, também executa um escopo operacional produzindo artefatos para serem enviados para a FS.

Os mecanismos propostos são:

- Cenários que serão utilizados nas iterações.
- Escopos de atuação da ODS e das FSs envolvidas nos cenários.
- Artefatos que deverão ser produzidos em cada escopo.
- Solicitação de Serviços (SS), como um instrumento de comunicação entre a ODS e uma FS.
- Ordem de Serviço Interno (OSI) como um instrumento de controle dos serviços realizados dentro da ODS pelas suas unidades organizacionais.
- Processo de controle das interações entre a ODS e a FS.
- Processo de qualidade que garantirá que os artefatos produzidos pela FS atende aos requisitos.

O comportamento interno da FS não será objeto de estudo neste trabalho. Todos os mecanismos definidos estão voltados para o ambiente da ODS.

Todas as abordagens feitas neste capítulo tomam como premissa básica que as equipes da ODS e da FS têm plena fluência na produção e na leitura de artefatos especificados em UML.

5.2 Análise dos Cenários

Para investigar as diversas possibilidades de interação entre uma ODS e uma FS, serão utilizados cenários. Um cenário, neste contexto, é a execução das

seis disciplinas[11] operacionais do RUP, feitas de forma colaborativa entre a ODS e a FS por ocasião de uma iteração.

Todos os cenários terão os seguintes atores:

- Empresa Contratante da Fábrica de Software (ECFS).
- Organização de Desenvolvimento de Sistemas (ODS) pertencente à ECFS.
- Fábrica de Software (FS).
- Usuários do sistema pertencente à ECFS.

A ODS para executar as disciplinas do RUP deverá utilizar uma estrutura organizacional que deverá ser tratada mais adiante no Capítulo 6.

A determinação do escopo da ODS e da FS em cada cenário, é uma função do gerente de projeto, definindo quais as disciplinas operacionais que cada organização deverá executar no cenário (iteração). Como cada disciplina do RUP tem suas atividades e cada atividade produz um determinado artefato (diagramas, modelos, documentos, etc), a definição das disciplinas e respectivas atividades executadas pela ODS irão indicar quais os artefatos da UML que deverão compor uma Solicitação de Serviço (SS) a ser enviada para a FS.

Os objetivos dessa seção são:

- Propor um modelo para a fundamentação das definições de cenários e escopos de atuação da ODS e da FS.
- Para cada cenário e respectivos escopos, propor um pacote de documentação que a ODS deverá entregar à(s) FS'(s), atendendo ao

[11] O termo "Execução da Disciplina" no presente trabalho significa a execução das atividades associadas à uma disciplina do RUP.

requisito de que essas fábricas deverão estar remotamente localizadas e que a dependência de informações por parte da FS deverá ser nula.

5.2.1 Mecanismos de Controle das Interações ODS x FS

Conforme demonstrado na Figura 7, o ciclo de vida de um projeto pode ter quatro[12] ou mais iterações, em que cada iteração possui um cenário[13], o qual agrega um ou mais escopos, formando um determinado arranjo. Por outro lado, um cenário poderá ser utilizado em mais de uma iteração, dependendo do planejamento feito pelo gerente do projeto.

Um escopo é uma agregação de disciplinas operacionais formando um determinado arranjo. A seção 5.2.2 “Definição das Possibilidades de Escopos” descreve a formação desses arranjos de disciplinas operacionais dos escopos.

Os escopos de um cenário são executados por uma ODS ou por uma ou mais FSs. Um dos principais aspectos do planejamento de uma iteração consiste da montagem do cenário de execução, ou seja, da escolha de escopos cuja soma das disciplinas operacionais dos escopos escolhidos, resulte nas seis disciplinas operacionais de uma iteração, sem repetições. A soma dos escopos sempre corresponderá ao tamanho do cenário em foco, que cobre toda uma iteração.

Segundo este modelo as disciplinas gerenciais do RUP não fazem parte dos escopos. Isso se deve ao fato de não serem variáveis no planejamento de uma iteração, porque, em todas as iterações, tanto a ODS quanto a FS, dentro dos seus domínios de atuação, sempre executarão as três disciplinas gerenciais (Gestão do Projeto, Gestão de Configuração e Mudança e Gestão do Ambiente).

---

[12] Um projeto pode ter, no mínimo, quatro iterações, porque o RUP tem quatro fases e uma fase deverá ter pelo menos uma iteração.

[13] O cenário corresponde a uma iteração porque ambos possuem a mesma abrangência, ou seja, possuem as mesmas disciplinas operacionais. Obviamente, uma iteração também envolve as Disciplinas Gerenciais, mas o foco nesta seção está voltado para as disciplinas operacionais.

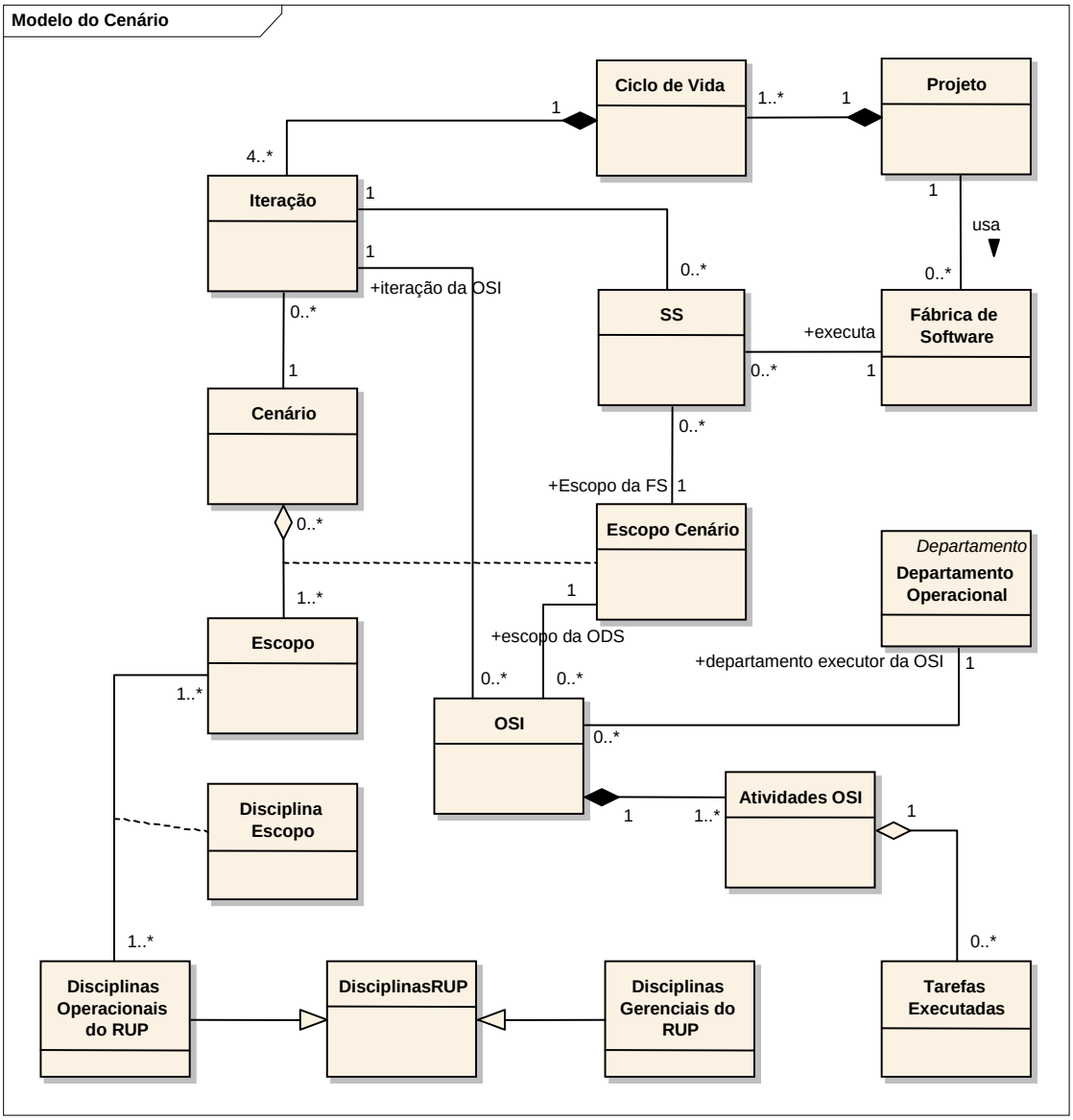


Figura 7 - Modelo dos mecanismos de controle das interações ODS x FS

A variedade de possibilidades de escopos (veja Tabela 1) permite uma grande flexibilidade na gestão do projeto por parte da ODS, pois, em cada iteração, o comportamento da ODS e da FS pode se alterar em função do escopo de cada uma (ODS e FS). Assim, a “costura” existente entre as duas organizações pode se alterar em cada iteração, dependendo dos escopos escolhidos para cada cenário.

A possibilidade de o cenário variar em cada iteração é plenamente possível do ponto de vista teórico, contudo, dentro de um contexto real, o mais comum é a utilização de um único cenário para todas as iterações das quatro fases do RUP [MEL2001].

O escopo a ser executado por uma FS (escopo FS) é considerado um serviço oferecido por ela. Dado uma iteração, esse serviço é realizado por uma única SS, ou seja, uma SS é usada para uma única iteração. Se uma outra iteração usar o mesmo cenário e aquele mesmo escopo for executado pela mesma FS, a ODS deverá elaborar uma outra SS.

O modelo (Figura 7) define que uma SS está associada a um único escopo da FS o qual é formado por um ou um grupo de disciplinas operacionais. Assim, pode-se afirmar que um escopo da FS (serviço) tipifica uma SS. Cada vez que uma SS é expedida, a FS deverá executar o serviço que corresponde à execução desse conjunto de disciplinas operacionais associadas ao escopo da FS em foco.

Simetricamente à FS, uma ODS também pode executar um escopo formado por uma ou mais disciplinas operacionais do RUP. Para a execução desse escopo, a ODS utiliza uma Ordem de Serviço Interno (OSI), a qual é semelhante à SS, porém com todo seu ciclo de vida realizado somente dentro da ODS. Por isso, ela precisará indicar todas as disciplinas e atividades que compõem o escopo a ser executado. Na SS, esse detalhamento das disciplinas envolvidas no escopo não ocorre, porque a ODS não controla o comportamento interno da FS. Ela só precisa ser informada do estado da SS ao longo da sua execução na FS.

É válido notar que o modelo permite que um projeto possa lançar mão de mais de uma FS. Isso permitirá ainda mais flexibilidade para a busca de uma FS mais especializada. O gerente do projeto pode distribuir as iterações para diferentes

FS, em função dos serviços que elas possam prestar, ou seja, dos possíveis escopos que elas possam executar, conforme definido na Tabela 1.

Com base nisso, pode-se concluir que não é a FS que tem escopo e sim o Serviço que ela presta para uma ODS. A FS existe por si só, com sua estrutura, recursos e tecnologias. Ela tem sua lista de serviços e está pronta para atender às SS que correspondam ao escopo do serviço necessitado pela ODS.

5.2.2 Definição das Possibilidades de Escopos

O objetivo desta seção é definir os possíveis arranjos entre as disciplinas operacionais do RUP para compor os possíveis escopos. Um ou mais escopos serão reunidos para compor um cenário a ser usado em uma iteração.

Analisando-se com mais detalhadamente a visão geral do RUP mostrado na Figura 1, pode-se perceber que, dentro de uma iteração, as disciplinas são executadas simultaneamente, porém, conforme demonstrado no “Plano de Iteração de Exemplo” [RUP2002], de cada fase do RUP, o início de cada disciplina se dá obedecendo a um certo seqüenciamento, constituindo-se em um fluxo operacional, conforme abaixo descrito:

1. Modelagem do Negócio (N).
2. Gestão de Requisitos (R).
3. Análise-*Design* (D).
4. Implementação (P).
5. Testes (T).
6. Implantação (I).

A matriz abaixo (Tabela 1) mostra que os arranjos entre as seis disciplinas operacionais do RUP correspondem às possibilidades de escopo de atuação da ODS e da FS.

Disciplinas	N	R	D	P	T	I	Totais
N	N						1
R	NR	R					2
D	NRD	RD	D				3
P	NRDP	RDP	DP	P			4
T	NRDPT	RDPT	DPT	PT	T		5
I	NRDPTI	RDPTI	DPTI	PTI	TI	I	6
Total de escopos	6	5	4	3	2	1	21

Tabela 1 - Possibilidades de escopos

Cada célula da matriz corresponde a um escopo formado por uma, ou por um conjunto de disciplinas operacionais. Assim, temos na célula formada pela coluna N e linha I (N x I) a definição do escopo **NRDPTI.** Isso significa que o escopo é formado pelo fluxo operacional que possui as seguintes disciplinas: **N** (Modelagem de Negócio), **R** (Gestão de Requisitos), **D** (Análise-*Design*), **P** (Implementação ou Programação), **T** (Testes) e **I** (Implantação).

Este é o escopo mais completo que uma organização (ODS ou FS) pode assumir, porque inicia na Modelagem de Negócio e termina na Implantação. Isso poderia se repetir em todas as iterações de todas as quatro fases do RUP. Se isso ocorrer, equivaleria ao que Fernandes [FER2004] classifica de Fábrica de Projetos Expandida.

A Tabela 1 mostra que existem 21 possibilidades de escopos, que podem ser executadas tanto pela ODS como pela FS. Do ponto de vista de uma FS, ela pode oferecer até 21 tipos diferentes de serviços, em que cada serviço corresponde a um escopo. Como qualquer dos escopos pode ser executado tanto pela ODS como por uma FS, logo uma ODS não precisa subcontratar o escopo mais extenso (NRDPTI), desde o levantamento de Requisitos até a Implantação. Segundo o modelo do cenário (Figura 7), o Gerente do Projeto

pode planejar os cenários de cada iteração das quatro fases compondo diferentes escopos em função dos recursos que a ODS dispõe no momento.

Para ilustrar uma fração das possibilidades que o gerente do projeto tem para fazer seu planejamento, será apresentado o seguinte caso:

> *O gerente do projeto tem uma equipe de analistas de negócio e analistas de sistemas que executam atividades das disciplinas de Modelagem de Negócio, Requisitos e Análise e Projeto. Em cada iteração, a ODS poderá lançar mão de uma ou de mais FS para executar um determinado escopo definido na Tabela 1.*

Desenvolvimento do Caso

- Fase de Iniciação:
  - ODS executa o escopo NRDPTI.
  - FS não participa da iteração da fase de Iniciação.
- Fase de Elaboração:
  - ODS executa o escopo NRD.
  - ODS elabora uma SS com o escopo P.
  - FS executa o escopo P.
  - ODS executa o escopo TI.
- Fase de Construção:
  - ODS executa o escopo NR.
  - ODS elabora uma SS com o escopo AP.
  - FS executa o escopo DP.
  - ODS executa o escopo TI.
- Fase de Transição:
  - ODS executa o escopo NRD.
  - ODS elabora uma SS com o escopo P.
  - FS executa o escopo P.

- o ODS executa o escopo TI.

A Figura 8 mostra as variações dos escopos da ODS e da FS em cada iteração para o caso acima apresentado. Os escopos tracejados são da ODS, e os pontilhados são da FS.

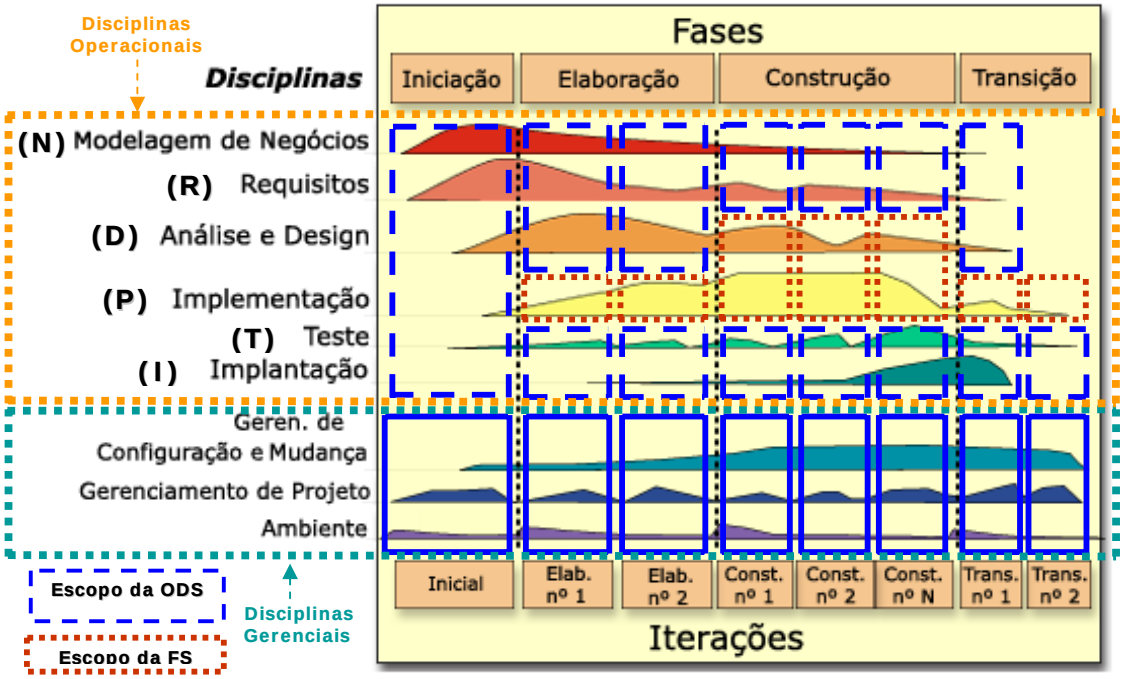


Figura 8 - Iterações com seus respectivos escopos da ODS e da FS (Adaptado do RUP [RUP2002])

A análise do caso acima mostra os seguintes fatos:

- Na fase de Iniciação, a ODS não utilizou a FS. Seu escopo foi o mais amplo previsto (NRDPTI).
- Nas fases de Elaboração, de Construção e de Transição, a ODS executou dois escopos, enquanto que a FS executou somente um.
- O processo de execução de uma iteração foi o seguinte:

    - ODS executa um escopo.
    - ODS elabora uma SS para a FS executar um determinado escopo.
    - FS executa o escopo indicado na SS.
    - ODS complementa a iteração executando mais um escopo (TI).
- O escopo complementar (TI) executado pela ODS que ocorreu em todas as interações de todas as fases que utilizaram a FS se justifica porque a ODS precisa testar e integrar os artefatos de software produzidos pela FS.
- Nas instâncias da fase de Construção, a ODS mudou a estratégia de utilização da FS, aumentando seu escopo de P para AP.
- Nada impede que o gerente diversifique a utilização de FS, entregando SS para mais de uma FS, buscando especialização, melhores preços e prazos.
- O gerente do projeto pode, opcionalmente, desenvolver uma estratégia especial para a fase de Construção em que as iterações sejam desenvolvidas em paralelo com uma, ou com múltiplas FSs. Como exemplo, supõe-se que a ODS tenha preparado em paralelo todas as iterações da fase de Construção, usando o escopo NR, e as tenha entregado a várias FSs para executarem o escopo AP. Isso irá proporcionar maior velocidade ao projeto, reduzindo significativamente o prazo de entrega. A única condição para a viabilização dessa estratégia é o plano de iterações. Do ponto de vista do processo indicado neste trabalho, não existe nenhuma restrição. Pelo contrário, é a característica iterativa que proporciona essa possibilidade do paralelismo.
- Na fase de Transição, a FS volta a ter um escopo bem reduzido (P), à semelhança da fase de Elaboração. Isso se justifica pelo fato de que o foco desta fase é a implantação do sistema como um todo no ambiente de produção da empresa proprietária do sistema em foco. As disciplinas de Teste e Implantação (TI) continuarão sendo executadas pela ODS. Caberão à FS somente as correções de falhas que não foram percebidas por ocasião das iterações, bem como a implementação de pequenas

mudanças que foram capturadas após os testes da última iteração da fase de Construção.

Na Figura 8, em cada iteração, as disciplinas gerenciais se encontram envoltas por um retângulo azul, indicando que é executado pela ODS. É valido salientar que, embora essas disciplinas sejam executadas em cada iteração, elas não se constituem em um escopo, porque todas as iterações executam as mesmas três disciplinas, tanto na ODS como na FS, não se constituindo em uma variável, e sim uma constante.

Como uma variante desse caso, vamos supor que a ODS não tenha recursos disponíveis para a execução de nenhuma das disciplinas operacionais, ou seja, a ODS não irá executar nenhum dos escopos definidos na Tabela 1. Essa variante obrigará o gerente de projeto lançar mão de uma ou mais FS que execute todas as disciplinas operacionais. Visando a mostrar os benefícios da estratégia gerencial orientada a escopo, será apresentado abaixo o seguinte planejamento (Figura 9):

- Fase de Iniciação (uma iteração):
    - ODS elabora SS para a FS1 com escopo NRDP.
    - FS1 executa o escopo NRDP.
    - ODS elabora SS para a FS2 com escopo TI.
    - FS2 executa escopo TI.
- Fase de Elaboração (uma iteração).
    - ODS elabora SS para a FS-3 com escopo NRDP.
    - FS-3 executa o escopo NRDP.
    - ODS elabora uma SS para a FS2 com o escopo TI.
    - FS2 executa o escopo TI.
- Fase de Construção (iterações).
    - Para cada iteração:
        - ODS elabora SS para a FS-4 com o escopo NR.

        - FS-4 executa o escopo NR.
        - ODS elabora uma SS para a FS-5 com o escopo AP.
        - FS-5 executa o escopo AP.
        - ODS executa uma SS para a FS2 com o escopo TI.
        - FS2 executa o escopo TI.
- Fase de Transição (uma iteração).
    - ODS elabora SS para a FS-4 com o escopo NRD.
    - FS-4 executa o escopo NRD.
    - ODS elabora uma SS para a FS-5 com o escopo P.
    - FS-5 executa o escopo P.
    - ODS executa uma SS para a FS2 com o escopo TI.
    - FS2 executa o escopo TI.

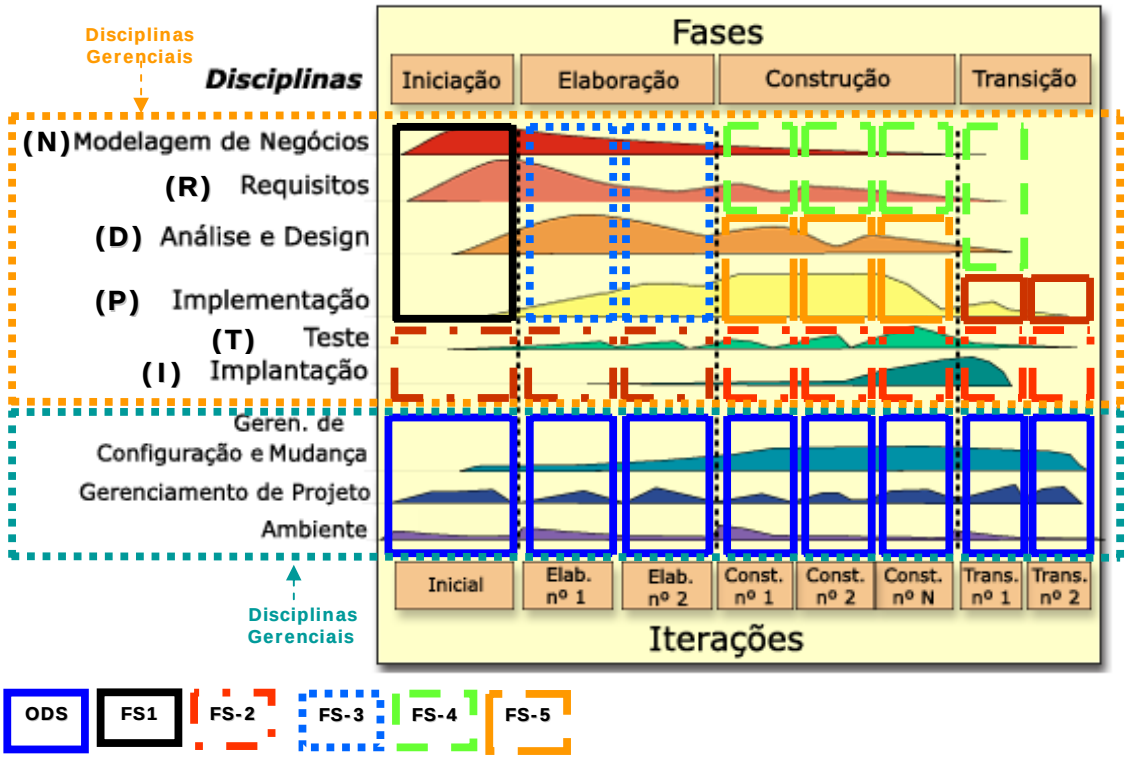


Figura 9 - Iterações com seus respectivos escopos de múltiplas FS (Adaptado do RUP [RUP2002])

A análise do caso com a ajuda da Figura 9 mostra os seguintes fatos:
A ODS utiliza cinco FS.

- A FS1 é especializada em Modelagem de Negócio, Gestão de Requisitos e elaboração de *Request for Proposal* (RFP). Na verdade, a FS1 nem precisa ser uma fábrica de software propriamente dita, mas uma empresa de prestação de serviços de RFP.
- A FS2 é especializada em teste e implantação de sistemas. Esse tipo de fábrica é conhecido como “Fábrica de Testes” [FER2004]. A FS2 é utilizada para executar o mesmo escopo em todas as iterações de todas as fases.
- A FS-3 é uma fábrica especializada em Arquitetura de Software. Possui profissionais altamente qualificados para a definição da arquitetura do aplicativo. Por isso só será utilizada na fase de Elaboração executando o escopo NRDP.
- A FS-4 é uma fábrica especializada em realização de casos de uso. Seu papel é detalhar os cenários de todos os casos de uso que não foram envolvidos na visão arquitetural de casos de uso na fase de Elaboração por ocasião da definição da Arquitetura. A FS-4 produz as especificações que serão envidas através da SS para a FS-5. Essa fábrica precisa ter bastante experiência em modelagem de projetos e UML para a criação de especificações de qualidade para que a FS-5 possa fazer a construção sem a necessidade de interagir com a FS-4.
- A FS-5 é uma fábrica de software especializada em construção de componentes. Equivale à “Fábrica de Código” [FER2004]. A FS-5 irá construir os componentes a partir dos modelos produzidos pela FS-4. As FS-4 e FS-5 são usadas nas fases de Construção e Transição.

5.2.3 Produtos dos Cenários

O objetivo desta seção é definir os artefatos que deverão ser produzidos a partir dos escopos realizados pela ODS ou por uma FS.

A Tabela 2 mostra todos os artefatos que podem ser produzidos em cada disciplina do RUP [RUP2002]. Como um escopo envolve uma ou mais disciplinas, os produtos a serem criados por um escopo são a soma dos artefatos produzidos por cada disciplina envolvida pelo escopo.

Tabela 2 - Artefatos produzidos nas disciplinas do RUP

Artefatos segundo o RUP [RUP2002]	Disciplinas Operacionais					
	Modelagem do Negócio	Requisitos	Análise e *Design*	Implementação	Testes	Implantação
Glossário de Negócio.	X					
Regras de Negócio.	X					
Modelo de Casos de Uso de Negócio.	X					
Modelo de Objetos de Negócio.	X					
Avaliação da Organização-alvo.	X					
Visão do Negócio.	X					
Documento de Arquitetura de Negócio.	X					
Especificação do Negócio.	X					
Realização dos Casos de Uso de Negócio.	X					
Plano de Gerenciamento de Requisitos.		X				
Glossário.		X				
Documento de Visão do Sistema.		X				
Modelo de Casos de Uso do Sistema.		X				
Especificações Suplementares.		X				
Atributos dos Requisitos.		X				
Protótipos da Interface com Usuários.		X				
Classes de Fronteira.		X				
Modelo de Análise.			X			

Modelo de *Design*			X			
Documento de Arquitetura do Sistema.			X			
Protótipo Arquitetural.			X			
Prova de Conceito Arquitetural.			X			
Arquitetura de Referência.			X			
Modelo de Dados.			X			
Modelo de Implementação.				X		
Componentes.				X		
Plano de Integração do *Build.*				X		
Plano de Testes.					X	
Sumário de Avaliação de Testes.					X	
Script de Testes.					X	
Lista de idéias de Testes.					X	
Casos de Teste.					X	
Modelo de Análise de Carga de Trabalho.					X	
Dados de Testes.					X	
Arquitetura para Automatização de Testes.					X	
Guia de Teste					X	
Configuração de Ambiente de Teste.					X	
Especificação da Interface de Teste.					X	
Conjunto de Testes.					X	
Classes de Teste.					X	
Componente de Teste.					X	
Plano de Implantação.						X
Lista de Materiais.						X
Notas de *Release*.						X
Produto.						X
Artefatos de Instalação.						X
Materiais de Treinamento.						X
Materiais de Suporte aos Usuários.						X
Unidade de Implantação.						X
Arte Final do Produto.						X

Tabela 2 - Artefatos produzidos nas disciplinas do RUP (Continuação)

5.3 Caracterização da ODS

5.3.1 Requisitos de Uma ODS-padrão

Esta seção objetiva descrever as características de uma ODS-padrão para ser usada na análise do comportamento dos cenários acima descritos. O conjunto dessas características define o conceito da ODS usada neste trabalho. As características são:

1. Domínio da UML para uma completa e correta produção de modelos, diagramas, metadados e especificações a serem enviadas para a FS.
2. Uso do RUP como modelo de processo para eliminar a dependência da experiência pessoal do gerente do projeto e vincular o sucesso do projeto ao controle do processo.
3. Processo formalmente definido para cada disciplina, mas que possa ser customizado para atender às especificidades de cada projeto.
4. Artefatos bem definidos e caracterizados para cada atividade de cada disciplina.
5. Processo de garantia de qualidade dos artefatos produzidos dentro da ODS e que deverão ser enviados para a FS.
6. Processo de recebimento e validação dos artefatos originários da FS visando a garantir a qualidade do que está sendo recebido.
7. Ter toda infra-estrutura de hardware e software necessária para a realização dos testes de sistema.
8. Disciplinas do RUP executadas por células de trabalho especializadas, compostas por profissionais altamente especializados nas atividades das disciplinas em foco.
9. Possuir uma equipe de métricas para fazer estimativas do porte dos escopos que estão sendo executados na ODS e que serão enviados para a FS. Esta equipe será a responsável por manter o histórico de eventos

de desenvolvimento que dê suporte à equipe de Planejamento e Controle da Produção (PCP) na estimativa de prazos.

10. *Frameworks* necessários para a implementação de arquiteturas de software para atender aos requisitos dos sistemas a serem desenvolvidos.
11. Infra-estrutura necessária para suportar o ciclo de vida dos projetos de desenvolvimento de software, inclusive ferramentas que possam maximizar a automação do processo visando à obtenção de produtividade.
12. Possuir uma estrutura necessária para suportar a gestão de componentes visando à maximização de reutilização.

Não faz parte deste trabalho a descrição das atividades relacionadas aos aspectos financeiros e administrativos da ODS, bem como o detalhamento de todas as características acima relacionadas. Só serão detalhadas aquelas que estiverem diretamente relacionadas com o objetivo deste trabalho.

5.3.2 Modelo da ODS

A Figura 10 mostra o modelo da ODS. Conforme demonstrado, ela é formada por departamentos que podem ser de caráter gerencial e de caráter operacional. Os departamentos gerenciais executam atividades relacionadas às disciplinas gerenciais do RUP, tais como: Gestão de Projetos, Gestão de Configuração e Mudança e Ambiente. Todos os escopos executados em um projeto, independentemente se for executado na ODS ou na FS, são planejados e controlados pelos departamentos gerenciais.

Os departamentos operacionais estão relacionados com as seis disciplinas operacionais do RUP. Esses departamentos se categorizam, ainda, em:

- Departamentos de Produção;

- Departamentos de Suporte à Produção.

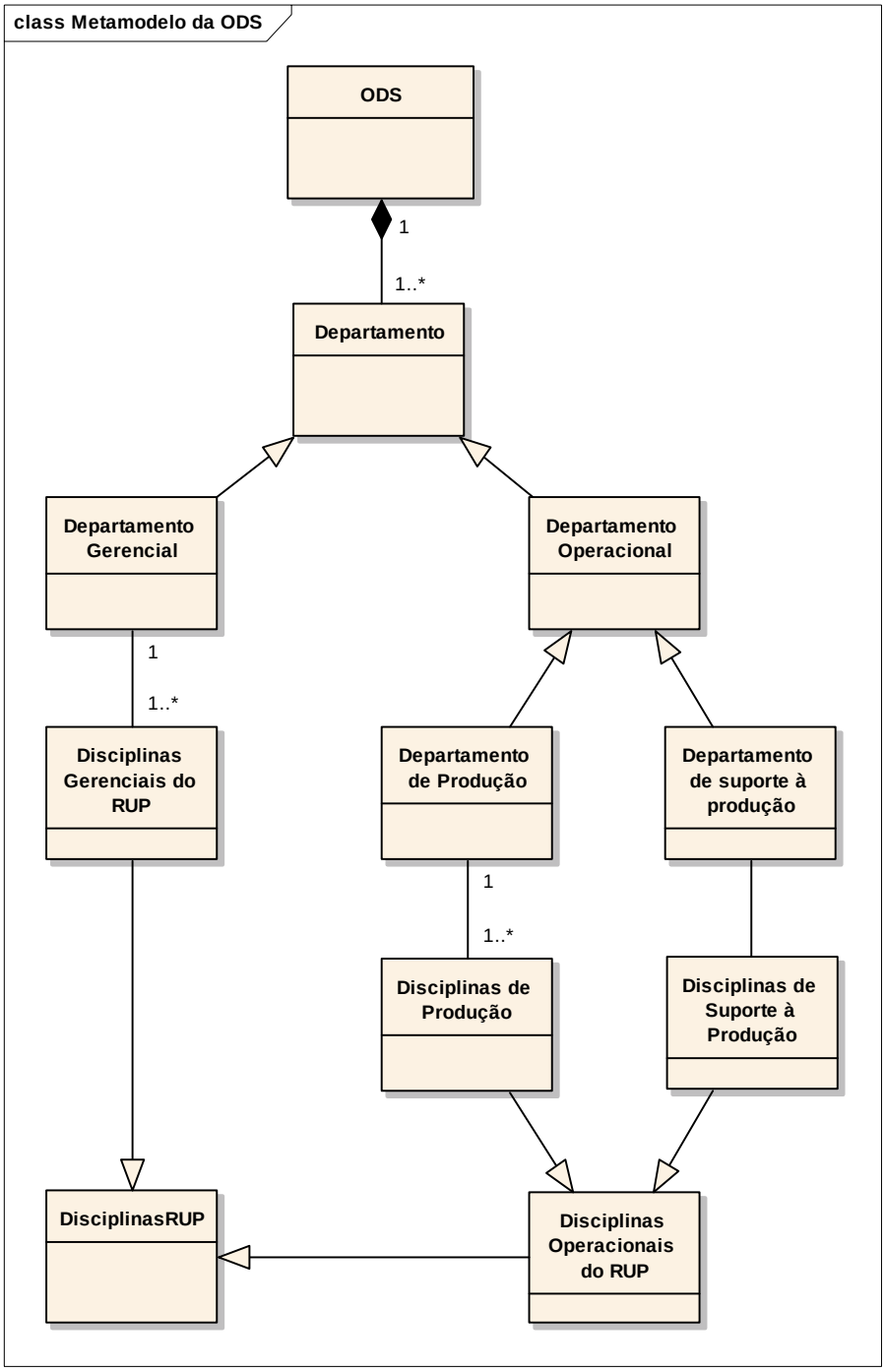


Figura 10 - Modelo da ODS

Os departamentos de produção são os responsáveis pela execução das disciplinas de Modelagem do Negócio, Requisitos, Análise-*Design* e Implementação. Eles produzem artefatos intermediários, (arquiteturas, modelos, diagramas, visões, itens da UML) e artefatos de software.

Os departamentos de suporte à produção executam as disciplinas de Teste e Implantação. Esses departamentos irão dar continuidade ao ciclo de vida dos produtos intermediários e produtos de software produzidos pelos departamentos de produção.

5.4 Processo Operacional

O objetivo dessa seção é a definição do processo operacional de controle das interações entre a ODS e as FSs envolvidas no processo produtivo de software. É válido salientar que a gestão deste processo fica na ODS. Isso quer dizer que as FSs envolvidas seguem o ritmo de trabalho definido pela ODS.

O comportamento da ODS depende de cada cenário, onde o conjunto de todos os cenários constitui um caso de desenvolvimento. O gerente do projeto assume o papel de orquestrador, compondo os cenários com os respectivos escopos, os quais indicarão o que, quando e quem produzirá todos os elementos do sistema.

A Figura 11 mostra um diagrama de atividades descrevendo um fluxo de controle da disciplina de Gestão de Projetos conforme definido no RUP [RUP2002]. É por meio desse fluxo que o gerente do projeto na ODS aciona uma, ou mais FSs para que elas produzam os artefatos associados ao escopo indicado na SS elaborada pela ODS. Esse produto pode ser um ou vários modelos previstos pelo RUP ou poderá ser um artefato de software devidamente testado pela FS.

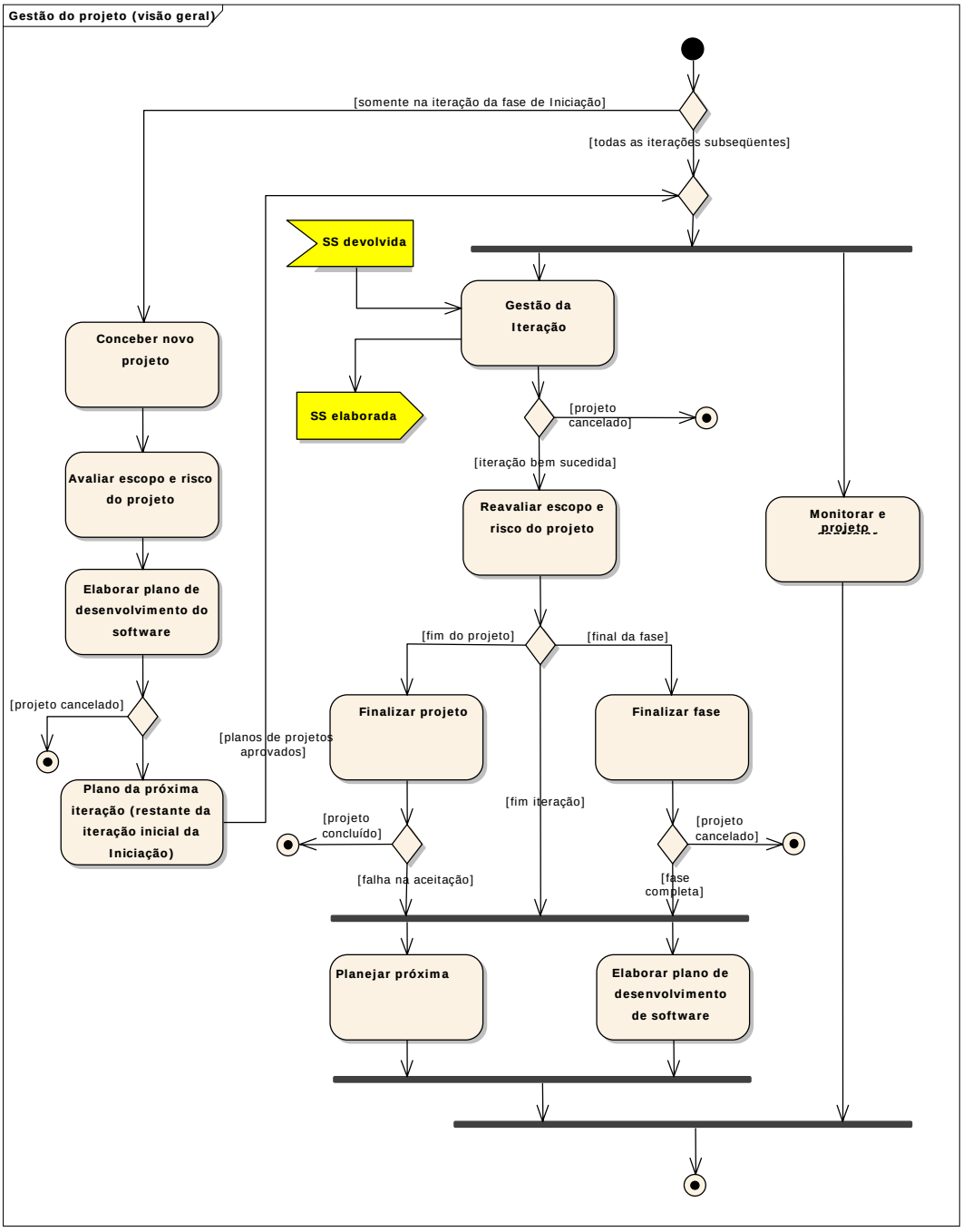


Figura 11 - Atividades da gestão de projetos (Adaptado do RUP [RUP2002])

No detalhamento do processo “Gerenciar Iteração” pertencente ao RUP [RUP2002], possui uma atividade “Iniciar Iteração” a qual tem uma tarefa chamada “Emitir Ordens de Trabalho” cujo objetivo é disparar a execução das disciplinas operacionais. É nesta ocasião que o gerente do projeto define qual o

cenário (definido na seção 5.2) que irá utilizar na iteração em foco. A decomposição funcional da tarefa “Emitir Ordens de Trabalho” constitui-se na execução do processo operacional proposto neste trabalho.

Por ocasião da execução do processo de Gestão de Projeto sugerido pelo RUP, uma SS poderá ser elaborada, fato que se constitui no evento (SS elaborada), mostrado na Figura 11, que irá disparar a execução do trabalho dentro da FS subcontratada.

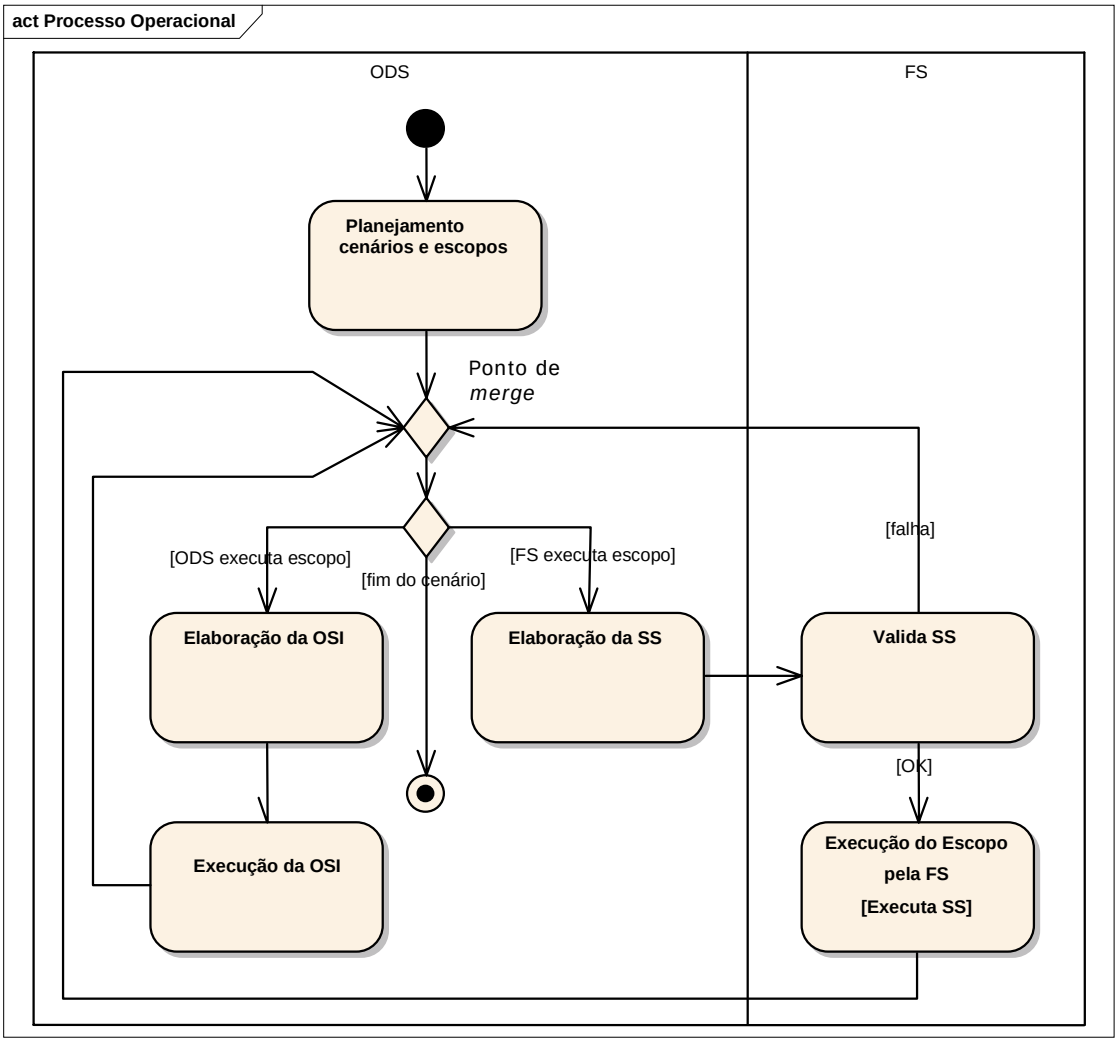


Figura 12 - Processo de controle das interações entre a ODS e FS

Caso a FS identifique falhas na SS por ocasião da sua recepção, também existe um outro evento de recepção para indicar os pontos de retorno da SS, originário da FS, para que seja corrigida e reenviada para a FS.

A Figura 12 mostra o detalhamento do processo operacional que gerencia as interações entre a ODS e as FSs envolvidas no cenário. Este processo consiste das atividades pertencentes às disciplinas operacionais que são executadas em um cenário de uma iteração constituído por um ou mais escopos de atuação da ODS e/ou FS.

O processo operacional não envolve as disciplinas gerenciais do RUP porque, conforme já comentado na seção 5.2 Análise dos Cenários, elas são constantemente executadas em paralelo com as disciplinas operacionais, constituindo-se em disciplinas guarda-chuvas. Por isso elas não variam em função das mudanças de cenários. O processo definido no RUP deverá ser executado normalmente sem nenhuma alteração.

Para demonstrar a execução do processo, serão apresentadas as execuções das fases de Iniciação e Elaboração referente ao caso ilustrado na Figura 8.

Na fase de Iniciação, o processo se comporta da seguinte forma:

1. O processo inicia na ODS por ocasião da tarefa “Emissão de Ordens de Trabalho”, que é uma primitiva funcional da atividade “Gerenciar Iteração” da disciplina de “Gerenciamento de Projetos do RUP” (figura 11).
2. O gerente de projeto tem recursos suficientes para o desenvolvimento de todo o processo operacional sem o suporte de FS. Por isso, durante a execução da atividade “Planejamento dos Cenários e Escopos” da figura 12, o gerente do projeto planeja um cenário com um único escopo (NRDPTI – veja tabela 1) para a iteração da fase de Iniciação.

3. O gerente do projeto elabora a OSI para ser executada por um departamento de produção da ODS.
4. O departamento de produção da ODS executa a OSI para produzir os artefatos característicos das disciplinas envolvidas no escopo indicado na OSI, após o que, o controle do fluxo retorna para o Ponto de *Merge*[14] logo após a atividade “Planejamento dos Cenários e Escopos”.
5. O gerente do projeto elabora uma nova OSI endereçada ao Departamento de Qualidade para que se proceda a verificação dos produtos criados pelo departamento de produção.
6. O departamento de qualidade executa a OSI e produz um relatório de verificação dos artefatos verificados.
7. Após a conclusão da validação de todos os artefatos produzidos pela completa execução do escopo, o gerente do projeto abrirá uma OSI para o “Departamento de Gestão de Configuração e Mudança” para a criação de uma *baseline* do pacote de produtos do escopo.
8. O fluxo de controle retorna para o Ponto de *Merge*, e a decisão será “fim do cenário”, porque todas as disciplinas operacionais já foram executadas na “Execução do Escopo da ODS”.

Na fase de Elaboração, o processo se comporta de forma diferente, porque o gerente de projeto irá utilizar uma FS para executar a disciplina de Implementação. A ODS irá executar todas as demais disciplinas operacionais, com exceção da Implementação. Com isso, a execução do processo operacional é feita da seguinte forma:

1. O processo se inicia na ODS executando o escopo NRD seguindo um fluxo semelhante ao acima exposto.
2. O controle do fluxo retorna ao Ponto de *Merge*, e a decisão deverá ser a de “FS executa escopo”. Com isso, o gerente de projeto irá executar a atividade de “Elaboração da SS”.

---

[14] Ponto de *Merge* é um ponto dentro do diagrama de atividades, no qual concentram-se todos os fluxos de retorno para reinício do processo.

3. A FS executa a atividade “Valida SS”.
4. Se a FS identificar alguma falha que possa impedir que a SS seja executada de forma independente da ODS, a SS é devolvida para a ODS. O fluxo retorna ao Ponto de *Merge*, e o gerente do projeto faz as devidas correções da SS e a reenvia para a FS. Se necessário, o gerente do projeto poderá restaurar a OSI executada, com o objetivo de corrigir possíveis falhas de artefatos que não foram identificadas pelas validações anteriormente efetuadas.
5. A FS, por sua vez, repete a atividade “Valida SS”.
6. Não tendo mais falhas, a FS faz a “Execução do escopo da FS (executa SS)”, que corresponde ao escopo **D** previsto na Tabela 1, após o que o fluxo retorna ao início do processo.
7. A próxima decisão a ser feita pelo gerente do projeto é a “ODS executa escopo” para realizar o escopo **TI** previsto no planejamento de cenários e escopos no início do processo.
8. Concluída a execução do escopo, o fluxo de controle retorna para o Ponto de *Merge*, encerrando o processo, porque é “fim do cenário”, ou seja, é fim do escopo.

Como pode ser notado, o processo é executado a cada iteração. Por isso, o processo permite que o planejamento do cenário possa ser retocado em cada iteração. É válido notar que a alteração dos escopos dos cenários ao longo do desenvolvimento nem sempre é possível, devido a aspectos contratuais e econômicos. Nota-se, ainda, que o processo possui flexibilidade para executar qualquer tipo de cenário previsto na seção 5.2 Análise dos Cenários.

5.5 Solicitação de Serviço (SS)

As seções 5.2.1 e 5.2.2, definiram os dois primeiros mecanismos (cenário e escopo) citados na Introdução do presente Capítulo. Esta seção definirá a SS como o terceiro mecanismo de controle entre a ODS e uma ou mais FS para a

produção de um sistema desenvolvido de forma iterativa, usando o RUP como modelo de processo de software e a UML como linguagem de notação.

Uma SS é um documento de mudança que formaliza a interação entre a ODS e uma FS por ocasião da execução de um único escopo em uma iteração. Ela é elaborada pelo Gerente do Projeto, o qual deverá indicar a data de emissão, data de envio para a FS, a identificação do escopo da FS com a indicação das disciplinas operacionais que deverão ser executadas e uma lista de artefatos, que seguirá anexa à SS. Deverá informar também uma lista de artefatos que deverão ser produzidos juntamente com um cronograma de entrega. A SS é enviada para a FS contratada para executar a SS.

A SS possui um ciclo de vida (Figura 13) que se inicia na ODS e depois acompanha todo o processo de execução dentro da FS, sendo finalizada quando retorna à ODS para que seja feita a validação do produto da SS.

A FS, ao receber a SS, faz uma verificação dos artefatos encaminhados, o escopo e o entendimento do serviço solicitado. Esta atividade se converte em um fator crítico de sucesso na validação das hipóteses, por isso é fortemente sugerido que ela seja feita com todo o cuidado possível, pois é por ocasião da recepção da SS que a FS poderá rejeitar as especificações recebidas por falta de clareza, completeza, padronização ou por qualquer outro motivo que possa dificultar o cumprimento do seu objetivo, que é produzir o resultado esperado pela ODS no prazo planejado e de forma desacoplada da ODS, conforme descrito na Hipótese-3, no Capítulo 1. Costa [COS2003] propõe um conjunto de atividades (padrões, métricas e técnicas consagradas pela Engenharia de Software) adaptado para a cultura brasileira. Não entraremos em detalhes na recepção da SS pela FS, porque não é foco do presente trabalho detalhar processos dentro da FS.

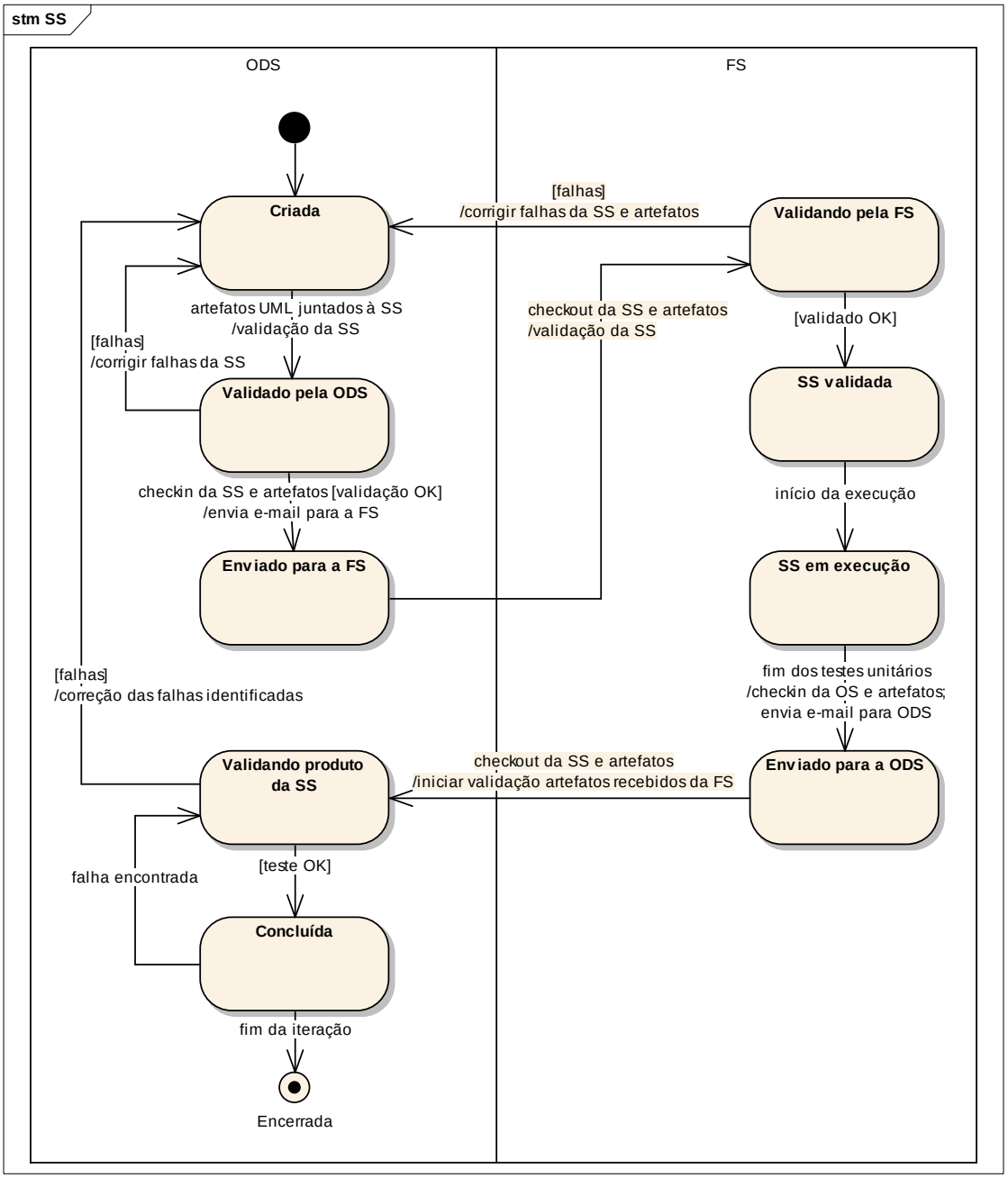


Figura 13 – Ciclo de vida da SS

A SS também tem o objetivo de registrar o histórico dos eventos ocorridos na comunicação ODS x FS. Converte-se numa oportunidade de registrar as realizações da equipe da FS com seus respectivos prazos previstos e realizados

para posterior utilização em estimativas. Isso se converte em temas para futuros trabalhos de pesquisa visando a responder às questões relacionadas a métricas.

Por ocasião do recebimento dos artefatos produzidos pela FS, a ODS deverá fazer uma validação do produto da execução da SS pela FS e registrar na SS a data de recebimento e os conceitos relacionados aos resultados da validação dos artefatos recebidos (software, modelos, arquiteturas, documentação complementar).

Sugere-se que a ODS disponibilize a SS em uma *extranet* para que haja constante transparência entre as duas organizações. Na medida em que um evento ocorra, este deve ser registrado na SS para que todos de ambas as organizações tomem ciência e as providências cabíveis. Infelizmente, isso não pode ser realizado no presente trabalho por falta de recursos disponíveis para a implementação dessa *Intranet*.

5.6 Ordem de Serviço Interno (OSI)

A OSI é um documento de mudança de uso interno da ODS que permite ao Gerente do Projeto solicitar a um Departamento da ODS que execute um determinado escopo. Enquanto que a SS tem um ciclo de vida no qual parte é executada na FS, e outra na ODS, a OSI tem todo seu ciclo de vida (Figura 14) realizado unicamente dentro da ODS.

Um escopo, em um cenário de uma iteração, ou é executado pela ODS ou por uma FS. Quando executado por uma FS, um escopo será associado a uma SS. Porém, quando um escopo é executado pela ODS, ele estará associado a uma OSI que corresponde uma ordem de serviço atribuída a um departamento da ODS para a execução de um escopo formado por uma ou mais disciplinas operacionais que serão executadas em uma iteração (figura 7). Uma OSI tem seu ciclo de vida totalmente executado dentro de uma iteração. Para a execução

do mesmo escopo da ODS em uma outra iteração, será necessária uma nova OSI.

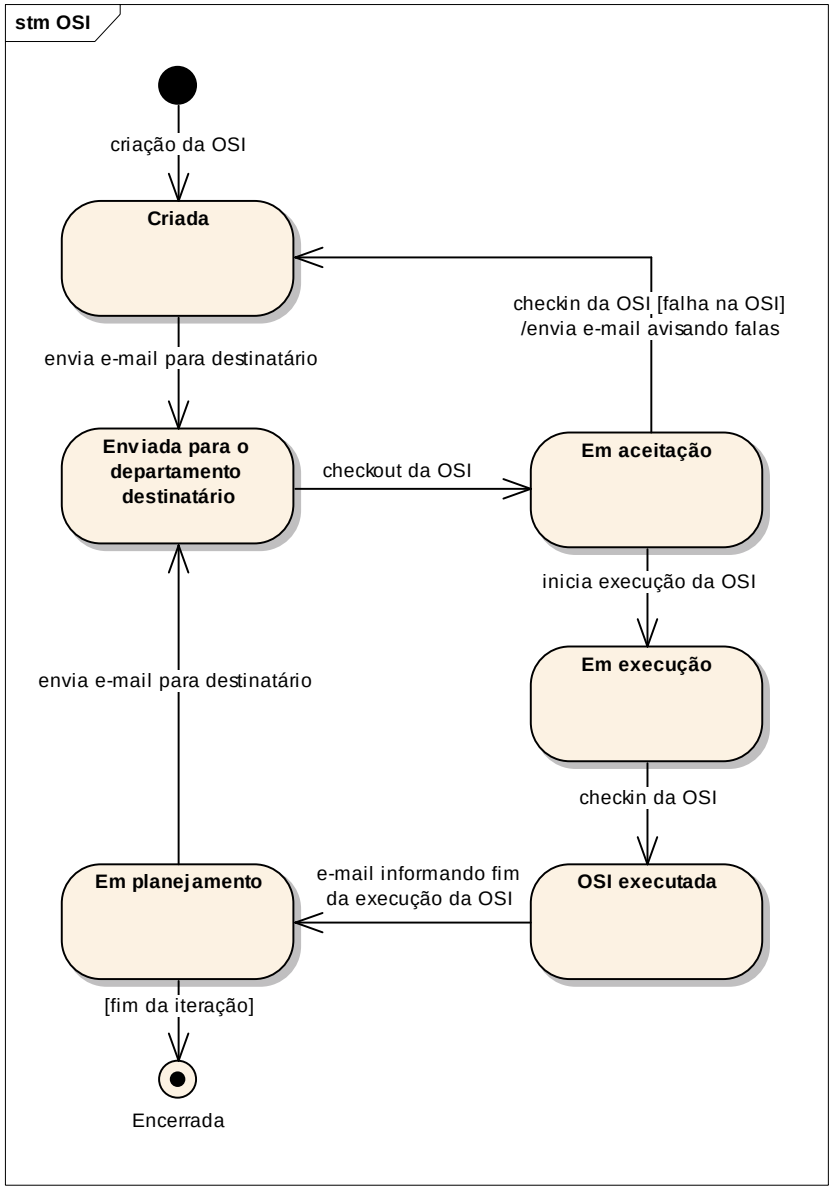


Figura 14 – Ciclo de vida da OSI

Por exemplo, se a ODS tiver que executar um escopo que tenha as disciplinas Modelagem do Negócio e Requisitos, bastará elaborar uma única OSI, porque o

departamento de Requisitos é especializado nestas duas disciplinas. Contudo se o escopo da ODS for **TI**, envolvendo as disciplinas de Teste e Implantação, então serão necessárias duas OSIs, porque a disciplina de Teste é uma especialização do Departamento Gestão da Qualidade, enquanto que a disciplina de Implantação é uma especialidade do Departamento de Implantação. Como se vê, são dois departamentos. Uma OSI não será suficiente porque ela é endereçada a um único departamento.

Com isso, o gerente do projeto assume o papel semelhante ao do regente de uma orquestra. Ele é quem comanda o início da execução de cada escopo, seja na ODS, seja nas FSs envolvidas. Se o escopo for realizado internamente, o evento estimulador será uma OSI. Se o escopo for executado por uma FS, o evento estimulador será uma SS.

5.7 Aspectos Econômicos no Planejamento dos Cenários

Embora não seja foco do presente trabalho mostrar a viabilidade econômica do uso dos mecanismos propostos, é válido salientar que o gerente do projeto deverá ter a preocupação de definir cenários que sejam factíveis, do ponto de vista prático e econômico.

Conforme demonstrado nas seções anteriores, do ponto de vista técnico, é plenamente possível o planejamento de cenários, em que cada iteração pode ter um cenário diferente, particionado em escopos para serem executadas por diferentes organizações de desenvolvimento, seja uma FS subcontratada ou qualquer outra organização interna ou externa geograficamente remota. Tecnicamente, a flexibilidade é total. Além disso, devido ao fato de a FS ser desacoplada, nada impede que, por motivo de força maior, ela possa ser substituída por outra, ainda dentro do ciclo de vida do projeto.

Contudo, essa flexibilidade é limitada pelo bom senso do gestor, pelo contrato celebrado com a FS no início do projeto e pelos limites impostos pelos aspectos comerciais entre essas organizações. Esses aspectos levarão o gerente a planejar cenários mais constantes ao longo das muitas iterações de todo o ciclo de vida do projeto, conforme sugerido por Mello [MEL2001].

Mello [MEL2001] sugere seis tipos de cenários que podem ser aplicados. Ele define uma estratégia horizontal, ou seja, todas as iterações possuem os mesmos escopos e não comenta a possibilidade de variá-los em diferentes fases. Em sua proposta, Mello permite que o controle do processo seja da FS, com a ODS assumindo um papel reativo, podendo executar algum escopo além das disciplinas gerenciais, porém sob o ritmo imposto pela FS subcontratada.

Fernandes [FER2004] prevê escopo de Fábrica de Componentes que tem o escopo “P” (Implementação). O escopo de Fábrica de Projeto proposto por Fernandes engloba as disciplinas de Análise-*Design* e Implementação, envolvendo a Fábrica de Componentes, diferente da proposta deste trabalho, que define um escopo restrito à disciplina de Análise-*Design* e um outro escopo restrito à disciplina de Implementação.

Bartie [BAT2006] apresenta os conceitos de Fábrica de Teste como uma alternativa que a ODS pode lançar mão para subcontratar as atividades relacionadas à disciplina de Teste (escopo de teste “T”). Uma pesquisa efetuada na Internet no dia 04/08/2007 mostrou mais de dez fábricas de teste no Brasil (não foram visitados todos os *web sites* encontrados na pesquisa), mostrando a viabilidade econômica para fábricas que procuram se especializar em escopos mais restritos e específicos.

Com base no exposto, conclui-se que a viabilização não se encontra no tamanho do escopo e sim na constância do escopo ao longo do ciclo de vida do projeto.

Capítulo 6 - Pesquisa-ação

6.1 Introdução

O objetivo deste capítulo é descrever os procedimentos tomados por ocasião da pesquisa-ação e apresentar os resultados obtidos visando a comprovar as hipóteses 3 e 4 apresentadas no Capítulo 1. O experimento foi realizado em uma empresa de consultoria em informática que possui uma estrutura organizacional matricial forte [PMI2004]. Dentre os projetos da empresa, foi escolhido um, cujo cliente situa-se no Rio de Janeiro, fato que permitiu o planejamento de cenários com três escopos, os quais serão descritos mais adiante. Este capítulo também mostrará a estrutura da ODS e das FS que foram simuladas no presente projeto, bem como os mecanismos e suas utilizações ao longo da pesquisa. Apresentará ainda todos os problemas, dificuldades encontradas ao longo do processo, as soluções tomadas em busca da comprovação das hipóteses 3 e 4 e os resultados finais encontrados. Não será foco desta pesquisa-ação estudar os processos dentro das FSs envolvidas no projeto, mas tão somente o refinamento dos mecanismos que uma ODS pode lançar mão para desenvolver sistemas de grande porte, com elevado grau de complexidade, usando um processo incremental e iterativo com uso de uma ou mais fábricas de software, também de forma iterativa, porém desacopladamente.

6.2 Estrutura Organizacional da ODS

A empresa que patrocinou a presente pesquisa possui uma estrutura organizacional baseada em projetos, em que cada projeto possui sua própria equipe. Isso foi considerado um problema para a empresa devido aos seguintes motivos:

- Falta de reutilização de componentes.

- Dificuldade de encontrar profissionais com um perfil abrangente, cobrindo as disciplinas de Modelagem do Negócio, Requisitos, Análise-*Design*, Implementação e Implantação. É mais fácil encontrar pessoas especializadas.

Em busca de uma solução para esse problema, a empresa decidiu mudar sua estrutura organizacional da área de desenvolvimento de sistemas para uma estrutura matricial forte [PMI2004]. A Figura 15 mostra esta estrutura formada por departamentos cujos papéis são aderentes à execução das disciplinas do RUP. A Tabela 3 mostra as nove disciplinas do RUP e os departamentos que se especializaram nessas disciplinas.

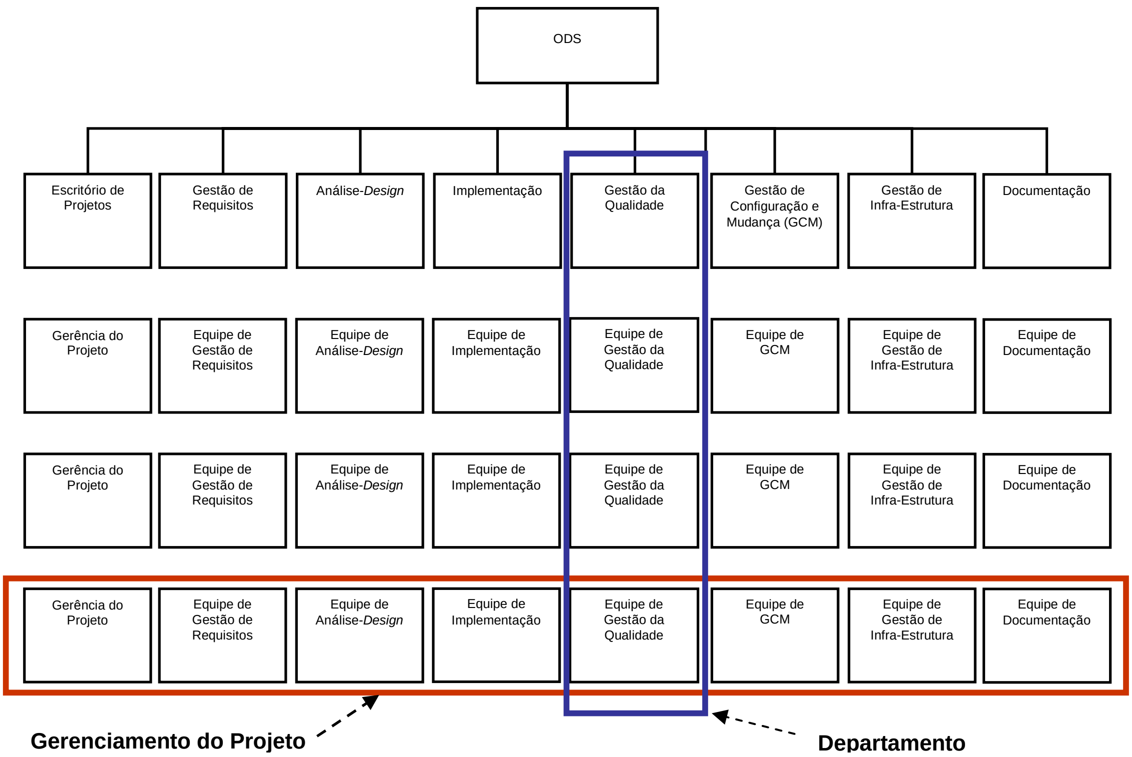


Figura 15 – Estrutura Matricial da ODS

As equipes dos projetos em andamento foram desfeitas, e os profissionais foram realocados nos departamentos da ODS, em função das suas especialidades.

Um corte horizontal da estrutura matricial, conforme apresentado pelo retângulo horizontal na Figura 15, mostra a forma como um projeto usa os diversos departamentos. Cada departamento, representado pelo retângulo vertical, atende a vários projetos, compondo uma equipe a partir dos profissionais alocados no departamento, de tal forma que um profissional pode atender a um ou mais projetos. Na medida em que o processo de desenvolvimento evolui, a equipe de cada departamento é utilizada de tal forma que a equipe total do projeto passa a ser a soma das equipes de todos os departamentos interceptados na horizontal da estrutura. Assim, pode-se dizer que um projeto é uma instância da ODS.

Disciplinas do RUP	**Departamentos da ODS**
Modelagem Negócio	Gestão de Requisitos
Gestão de Requisitos	Gestão de Requisitos
Análise-*Design*	Análise-*Design*
Implementação	Implementação
Testes	Gestão da Qualidade
Implantação	Gestão de Configuração e Mudança
Gestão do Projeto	Escritório de Projeto
Gestão de Configuração e Mudança	Gestão de Configuração e Mudança
Gestão do Ambiente	Escritório de Projetos e Gestão de Infra-Estrutura

Tabela 3 - Disciplinas do RUP x Departamento da ODS

Para compor a equipe total do presente projeto, foram contratados novos profissionais para atuarem nos departamentos de Gestão de Requisitos, Análise-*Design* e Implementação. Os demais departamentos participaram do projeto usando membros da equipe já existente.

Embora o projeto tenha se utilizado de toda a infra-estrutura da ODS, as equipes dos departamentos de Gestão de Requisitos, Análise-*Design* e Implementação usadas na pesquisa foram estrategicamente isoladas das demais equipes para

constituir uma instância especial e experimental da ODS. As diferenças dessa instância da ODS para as demais instâncias são as seguintes:

As equipes de Gestão de Requisitos e Análise-*Design* receberam novos membros com o perfil adequado para o projeto usado na presente pesquisa-ação. A equipe de Gestão de Requisitos ficou sediada no Rio de Janeiro, porque o cliente, dono do projeto a ser desenvolvido, situa-se naquela cidade.

Essa instância da ODS não conta com os departamentos de Análise-*Design* e Implementação porque eles assumiram os papéis de FS1 e FS2, respectivamente. Essas duas FSs também receberam uma equipe totalmente nova, com um perfil alinhado com as necessidades de uma FS que desenvolve sistemas OO, com uso do RUP e UML.

A comunicação entre os departamentos dentro da ODS foi permitida e incentivada, porém a comunicação entre os departamentos da ODS e as duas FS (FS1 e FS2) foi totalmente eliminada (para assuntos do projeto).

A segregação física da equipe de Gestão de Requisitos ocorreu por estarem trabalhando no Rio de Janeiro. Em São Paulo, a separação entre as equipes de Análise-*Design* e de Implementação, FS1 e FS2 respectivamente, não pôde ocorrer fisicamente, a exemplo da equipe de Gestão de Requisitos. A separação ocorreu de forma lógica por um processo que agiu como se as equipes estivessem geograficamente separadas. Além disso, foram definidas normas internas de comunicação entre as equipes, o que facilitou a gestão da comunicação dentro do projeto, internamente na ODS e na interação com as duas FSs.

A FS1 utilizou-se de profissionais recém-contratados e alocados no departamento de Análise-*Design*, executando todas as atividades da disciplina de Análise-*Design* do RUP. A FS2 também utilizou profissionais recém-

contratados e alocados no departamento de Implementação, executando todas as atividades da disciplina de Implementação do RUP, equivalente à Fábrica de Componentes sugerida por Fernandes [FER2004]. Com isso, a equipe de Análise-*Design* transformou-se na FS1, e a equipe de Implementação transformou-se na FS2.

A grande vantagem dessa estrutura organizacional é que se obtém otimização das equipes, maior produtividade, redução de custos e uma economia de escopo [GRE2004], permitindo um melhor gerenciamento dos componentes construídos no sentido de reutilizá-los em novos projetos.

6.2.1 Escritório de Projetos

Escritório de Projetos é uma unidade organizacional que centraliza e coordena o gerenciamento de projetos sob seu domínio [PMI2004]. Os projetos não têm equipes para executar as atividades das disciplinas operacionais. As equipes ficam dispersas nos departamentos especializados. O gerente do escritório de projetos designa um gerente de projeto para coordenar o processo de desenvolvimento. O papel do gerente de projeto é orquestrar o uso de cada Departamento no momento adequado para que cada um execute seu trabalho no projeto. O gerente do projeto não controla a execução dos procedimentos operacionais dentro de cada departamento da ODS nem dentro de uma FS.

6.2.2 Gestão de Requisitos

É um departamento de produção de artefatos responsáveis pela execução das disciplinas de Modelagem do Negócio e Requisitos. Produz todo o detalhamento necessário e suficiente dos requisitos e casos de uso para que o departamento de Análise-*Design* possa iniciar seu trabalho.

6.2.3 Análise – *Design*

É um departamento de produção de artefatos diretamente relacionado à disciplina de Análise-*Design* do RUP. Seu objetivo é fazer a realização dos

casos de uso originários do departamento de Gestão de Requisitos, produzindo o Modelo de Análise, Modelo de *Design*, Modelo de Implementação, Modelo de Implantação e Modelo de Dados com todo o detalhamento necessário e suficiente para que o departamento de Implementação, ou uma FS, faça a codificação dos componentes do sistema.

6.2.4 Implementação

É o departamento diretamente relacionado com a disciplina de Implementação do RUP. Seu objetivo é fazer a codificação e testes unitários dos componentes construídos. Em busca de uma maior produtividade, o departamento de Implementação foi dividido em duas células especializadas: construção de interfaces gráficas com o usuário e construção de componentes de serviço e de persistência de dados.

6.2.5 Garantia de Qualidade

É o departamento responsável pela disciplina de testes do RUP. Visando a oferecer maior produtividade, esse departamento foi dividido em duas células:

- Planejamento de qualidade definindo os casos de teste[15] por ocasião da elicitação dos casos de uso e da elaboração dos modelos de análise e de projeto.
- Aplicação das revisões por pares dos artefatos intermediários e testes de verificação dos produtos de software desenvolvidos pela FS envolvidas no processo.

A divisão em duas células se deve ao fato de o planejamento da qualidade e a aplicação dos testes ocorrerem simultaneamente, além de que essas duas atividades possuem elevado volume de trabalho, impedindo que uma mesma equipe possa dar conta das duas.

---

[15] Casos de teste são itens de teste de um produto de software [RUP2002]. Mede aspectos relacionados aos requisitos percebidos externamente pelos usuários [ISO9126].

6.2.6 Gestão de Configuração e Mudança

É o departamento que executa duas disciplinas do RUP: Gestão de configuração e mudança e Implantação. A gestão de configuração e mudança é uma disciplina “guarda-chuva” que acompanha paralelamente todo o processo de desenvolvimento, dando suporte na criação de *baselines*, versionamento e no controle das mudanças ocorridas. No que diz respeito à Implantação, executa todas as atividades que promoverão a entrega do aplicativo, ou seja, a instalação deste no ambiente de produção do cliente.

6.2.7 Infra-estrutura

Responsável pela gestão de toda a plataforma de hardware e software bem como pelo planejamento de capacidade visando a nunca faltar recursos para as equipes, contribuindo para a manutenção da produtividade. É um departamento que dá suporte à disciplina de Gestão do Ambiente executada no Escritório de Projetos.

6.2.8 Documentação

É um departamento que se responsabiliza em manter em ordem toda a documentação do sistema. Seus objetivos são:

- Garantir a qualidade dos metadados do sistema.
- Garantir que os modelos/diagramas se encontram coerentes e obedeçam ao padrão de organização na ferramenta CASE.
- Produzir os manuais de especificação do sistema, implantação, produção e usuário, todos dentro dos padrões previamente acertados com o Cliente.

Com essa nova estrutura, os funcionários existentes foram realocados nos novos departamentos, e novos funcionários foram contratados para complementar os papéis ainda vagos.

6.3 Estrutura da FS1 e FS2

As duas fábricas de software (FS1 e FS2) usadas na pesquisa foram simuladas, usando profissionais que foram contratados para este projeto, os quais foram alocados nos departamentos de Análise-*Design* e Implementação. A FS1 assumiu o papel do departamento de Análise-*Design*, enquanto que a FS2 assumiu o papel do departamento de Implementação.

A FS1 utilizou-se de um Arquiteto de Software com experiência em toda a tecnologia Java, UML, RUP, *design patterns*, especialmente na elaboração de arquitetura de software com visões arquiteturais recomendadas por Jacobson [JAC1999]. Além disso, acumulou a função de liderança da FS1. Além do arquiteto, a FS1 contou ainda com dois Projetistas para a elaboração do modelo de *design* do sistema foco da pesquisa, sob a orientação do Arquiteto acima referenciado.

A FS2 também foi também formada por profissionais contratados especificamente para o projeto foco desta pesquisa. Dois papéis foram utilizados na FS2: Analista-programador e o Programador. O Analista-programador tem um papel muito popular no Brasil e se caracteriza por um programador com bastante experiência na plataforma de software (Java, ferramentas e *ferameworks*) utilizado no projeto, além de possuir conhecimentos em modelagem da implementação com notação UML e em arquitetura de software. No presente projeto, este papel substituiu o Arquiteto de Software, sugerido pelo RUP [RUP2002]. Para desempenhar esse papel, foi contratado um único profissional, o qual assumiu a liderança da FS2. O segundo papel usado na FS2 foi o Programador, que equivaleu ao Implementador sugerido pelo RUP. O perfil exigido foi o conhecimento em UML e na plataforma Java. Para o papel de Programador, foram contratados seis profissionais, dois seniores, dois plenos e dois juniores.

6.4 Gerenciamento da Comunicação

O assunto comunicação na presente pesquisa abrange quatro aspectos:

- A SS como instrumento de comunicação entre a ODS e a FS.
- A OSI como instrumento de comunicação entre a Gerência do Projeto e a equipe interna do projeto descentralizada nos diversos departamentos da ODS.
- O processo de configuração e mudança como um intermediário comum a todos os departamentos da ODS.
- O processo de qualidade que intermediava o recebimento e envio de artefatos entre ODS e FS.
- Arquivo EAP produzido pela ferramenta de modelagem EA.
- A UML como notação a ser usada nos dois lados, ODS e FS.

6.4.1 A SS Como Instrumento de Comunicação

A SS, conforme já descrito na seção 5.5 deste trabalho, foi um importante instrumento usado na comunicação entre a ODS e uma FS. Sua função foi empacotar um conjunto de artefatos para serem usados como insumos por uma FS destinatária e ser um repositório dos registros dos eventos ocorridos por ocasião do processo produtivo dentro da FS e dos processos de qualidade e de implantação dentro da ODS.

6.4.2 A OSI Como Instrumento de Comunicação

A OSI assumiu um papel semelhante à SS, porém com todo seu ciclo de vida atuando somente dentro da ODS, registrando informações referentes aos pontos de controle dentro do processo de produção (nos departamentos de produção) e nos processos de qualidade, configuração e implantação.

6.4.3 O Processo de Configuração e Mudança Como Instrumento de Comunicação

O processo de configuração e mudança se converteu num importante instrumento para o estanqueamento dos departamentos da ODS e das FS envolvidas no processo. Embora a ODS tenha um departamento especializado na gestão de configuração e mudança, este processo se estende além das fronteiras desse departamento, permitindo a migração de artefatos entre os departamentos da ODS. A seção 6.5.1 faz o detalhamento deste processo.

6.4.4 O Processo de Qualidade Como Instrumento de Comunicação

Seu escopo foi bastante extenso, iniciando no planejamento dos testes e listas de verificações e se estendendo até a aplicação das verificações dos produtos intermediários [ISO9126] e testes dos produtos de software em cada iteração de todo o projeto. A execução do processo foi totalmente em paralelo às atividades das disciplinas de Modelagem do Negócio, Requisitos, Análise-*Design* e Implementação. Por isso, também foi considerado como um processo “guarda-chuva”. Foi o grande responsável pela geração de todo o *feedback* para todas as equipes da ODS e das duas FSs, fato que contribuiu de maneira importante para o melhoramento da qualidade interna dos artefatos e, conseqüentemente, dos produtos de software.

6.4.5 Arquivo EAP Produzido pelo EA

O EA é uma ferramenta de modelagem de software que usa notação UML, que tem um banco de dados que armazena todos os metadados do sistema em desenvolvimento, fato que permite a criação de todos os relatórios-padrão sugeridos pelo RUP. Para dar mais produtividade ao projeto, esses relatórios foram suprimidos porque o EA possui um recurso pelo qual múltiplas equipes podem trabalhar em artefatos diferentes de forma simultânea. A base central do EA estava instalada em um servidor de arquivos acessado por todas as equipes, inclusive as duas FSs. Como todas as especificações de todos os produtos

intermediários estavam descritas unicamente no EA, inclusive os casos de teste, isso veio a facilitar de forma importantíssima a comunicação (lícita) entre todas as equipes, porém com a supervisão do processo de configuração e mudança, descrito mais adiante neste trabalho.

6.4.6 A UML Como Instrumento de Comunicação

Todos os artefatos produzidos em um departamento da ODS ou pela FS, sempre foram descritos em UML. A única exceção se fez com o Modelo de Dados para o qual se utilizou a notação usada por Martin [MRT1990]. A decisão pela utilização dessa notação se deve ao fato da ferramenta de modelagem UML usada no presente projeto ainda não dar suportar a todos os passos de um projeto de banco de dados, além da popularidade da notação de Martin usada em todo o mundo.

A presente pesquisa tomou como princípio que a UML é uma linguagem universal de modelagem de software e que, se a ODS e a FS tiverem profissionais com “fluência” em UML, a comunicação entre essas duas organizações se dará de forma completa. Foi para esse propósito que a UML foi criada e tem sido aperfeiçoada por meio das novas versões. O fator crítico de sucesso para uma plena comunicação via UML é ter a garantia de que os profissionais utilizados em ambas as organizações tenham fluência no uso da notação. Assim, se um artefato UML for produzido pela ODS, certamente, será plenamente lida e compreendida por um outro profissional de uma FS geograficamente separada da ODS e sem nenhuma comunicação adicional com ela.

Na presnte pesquisa a UML mostrou ser como uma linguagem consistente e eficaz. Porém, para que a comunicação se faça fluentemente, é necessário que ambas as organizações (tanto a que produz o artefato quanto a que se utiliza dele) tenham pleno conhecimento de todos os elementos e sintaxe da UML para

que esta possa ser usada sem vícios e, assim, a comunicação seja estabelecida como o esperado.

A UML foi o mecanismo usado que mais necessitou de cuidados na presente pesquisa-ação. Abaixo, encontram-se alguns problemas que ocorreram e que requereu atenção especial para que o projeto alcançasse seu objetivo.

- A UML é uma linguagem de notação bastante comentada na mídia especializada, porém ainda pouco utilizada em sua plenitude. Existe ainda uma grande carência de profissionais que tenham o conhecimento suficiente para a produção e leitura de artefatos especificados com UML. Para a contratação de profissionais para o presente projeto, foi contratada uma consultoria para a seleção de profissionais no mercado com nível sênior em UML. O resultado foi que os profissionais encontrados não tinham senioridade em UML. Passou-se, então, a buscar profissionais ocupados em projetos em andamento. O fato é que esses profissionais cobravam muito acima dos valores médios de mercado. Para resolver o problema, foram contratados profissionais com nível pleno de experiência em UML.
- Logo no início dos trabalhos, percebeu-se que todos os profissionais contratados possuíam vícios que impediam a produção de artefatos com a especificação adequada. A solução para esse problema foi a aplicação de um treinamento (noturno) de quarenta horas em UML para o nivelamento do conhecimento.
- Mesmo assim, o nível desejado ainda não foi atingido pela equipe. A solução adotada foi o Gerente do Projeto, por ter vasta experiência em UML, acumular o papel de mentor. Ele conduziu as equipes aplicando um processo que se constituiu de rápidos treinamentos (*on-the-job*), especificamente planejados e aplicados a uma equipe de trabalho, para solucionar um problema bem definido, com a imediata aplicação da teoria no projeto com a supervisão do mentor. Na medida em que a equipe ia

adquirindo experiência e produtividade, o mentor ia se afastando da equipe e se dedicando a outras equipes, usando o mesmo método.

- Um outro problema que ocorreu e de difícil solução foi a falta de habilidade de redação por parte dos analistas da equipe na ocasião da descrição dos casos de uso, classes, atributos, operações e outros itens da UML. As revisões por pares dos artefatos conduzidas pela equipe de qualidade identificaram elevadíssima quantidade de descrição imprecisas, redundantes e principalmente ambíguas. Para a solução desse problema foi necessária a intervenção do Gerente de Projeto entrevistando pessoalmente cada analista e mostrando pontualmente as falhas identificadas nos artefatos verificados.

Resolvido o problema de linguagem de comunicação, resta garantir quais os artefatos mais adequados que deverão ser usados junto com uma determinada SS. Por isso, faz-se necessário que se estabeleçam quais os artefatos que devem ser produzidos na passagem de uma disciplina para outra em uma iteração. A seção 6.5.3 irá demonstrar os artefatos que foram produzidos em cada disciplina.

Embora o RUP recomende uma série de documentos, tais como Documento de Visão, Documento de Arquitetura, Relatório Sintético de Casos de Uso, Especificação de Casos de Uso, e outros, como já dito anteriormente, no presente projeto, nenhum desses documentos foi produzido. Isso foi motivado pelo uso do EA, o qual permitiu que trabalhos paralelos fossem realizados e depois integrados em uma única base centralizadora de todo o projeto. Essa integração foi realizada pela equipe do departamento de gestão de configuração e mudança. Quando uma equipe concluía seu trabalho, o responsável pela equipe fazia o *checkin*[16] no CVS e passava um e-mail para o líder do departamento de Gestão de Configuração e Mudança com cópia para o Gerente do Projeto. Um analista de configuração e mudança fazia o sincronismo dos

[16] *Checking* é o ato de incluir um artefato em um software de controle de versão.

arquivos no EA e atualizava a base de metadados contendo todos os modelos atualizados e íntegros. Com isso, o departamento de Configuração e Mudança se transformou em um instrumento crítico na integração dos produtos gerados pela ODS e pelas duas FSs.

No início dos trabalhos, a comunicação entre todos os departamentos da ODS sempre era por meio de uma OSI. Porém, isso ficou muito burocrático, e foi, logo no início, retirado do processo. Para dar velocidade nas comunicações, os departamentos se comunicavam por e-mails, sempre com cópia para o Gerente do Projeto. No entanto, se houvesse a necessidade de passagem de algum artefato de uma equipe para outra, isso sempre era feito pelo processo de configuração e de mudança.

6.5 Processo Utilizado

O processo operacional mostrado na figura 12 foi usado na ODS por ocasião da pesquisa-ação. Este processo utilizou dois outros processos suplementares, os quais são:

- Processo de configuração e mudança.
- Processo de qualidade.

6.5.1 Processo de Configuração e Mudança

O processo de configuração e mudança foi utilizado por todos os departamentos da ODS como um instrumento de comunicação. Nenhum documento foi passado diretamente de um departamento para outro. Sempre que um artefato era concluído, o responsável por aquele artefato fazia sua inclusão (*checkin*) no CVS. Com isso, nenhum departamento necessitou entregar diretamente um artefato para outro departamento. Essa transição de artefatos sempre foi feita pelo processo de configuração e mudança.

Todo o processo se baseia nos seguintes fundamentos:

- Toda e qualquer mudança requer uma SS ou uma OSI.
- Uma SS ou uma OSI acompanha todo o ciclo de vida da mudança, até que os artefatos produzidos estejam validados pelo departamento de qualidade da ODS.
- A SS e a OSI são documentos que registram todos os eventos ocorridos ao longo do processo de mudança.
- O processo de mudança é concluído após a validação dos artefatos produzidos na mudança e após a criação de uma *baseline* para eles.
- Todo artefato criado por meio de uma SS ou de OSI deverá ser incluído no CVS para controle de versão. Essa inclusão deverá ser feita pelo processo de *checkin.*
- Após a conclusão do processo de mudança, ou seja, após a criação da baseline, nenhuma alteração poderá ser feita no artefato versionado. Caso haja uma nova mudança que afete o artefato versionado, um novo processo de mudança precisará ser iniciado por meio da criação de uma nova SS ou OSI.
- Ao longo do processo de mudança, sempre que um artefato necessite ser envidado de uma unidade organizacional para uma outra unidade (interna ou externa à ODS), isso deverá ser feito por meio dos procedimentos de *checkin* e de *checkout.* A unidade organizacional remetente faz o *checkin* do artefato no CVS e envia um e-mail para o destinatário. O destinatário, por sua vez, faz o *checkout* dos artefatos e excuta seu trabalho.

O processo de configuração e mudança se encontra mais bem definido nas máquinas de estado que descrevem os ciclos de vida da SS (figura 13) e da OSI (Figura 14), bem como no processo operacional (Figura 19) após o refinamento obtido por meio da pesquisa-ação.

6.5.2 Processo de Qualidade

O processo de qualidade no presente projeto foi executado pelo Departamento de Gestão de Qualidade, e seu papel foi validar todos os artefatos produzidos pelos departamentos de produção e pelas FSs envolvidas no projeto. O processo de qualidade foi dividido em três segmentos:

- Planejamento da qualidade.
- Aplicação das verificações[17] e validações[18] dos produtos intermediários.
- Testes dos produtos de software originários da FS2.

O planejamento da qualidade tratou da:

- criação de listas de revisão (*checklist*) dos produtos intermediários;
- criação dos casos de teste para serem usados nos testes dos produtos de software.

As listas de revisão [SOF2006] (Anexo C) dos produtos intermediários foram elaboradas e incluídas no Guia de Qualidade de Software, o qual foi distribuído a todos os departamentos da ODS, inclusive às duas FSs. Cada lista de revisão era composta por:

- itens de revisão que abordavam aspectos técnicos que deveriam ser verificados pela equipe interna da ODS;
- Itens de revisão que abordavam aspectos relacionados aos requisitos, regras e conceitos de negócio que precisavam ser validados pelos usuários.

---

[17] Verificação é um tipo específico de teste de software que garante que um artefato foi implementado da melhor forma possível. Responde à pergunta "estamos construindo o produto corretamente?" [PRE2002]. As verificações foram realizadas pela equipe interna da ODS.

[18] Validação é um tipo específico de teste de software que garante que o artefato implementa o requisisto do cliente. Responde à pergunta "estamos construindo o produto certo?" [PRE2002]. As validações dos produtos intermediários foram realizadas pelos usuários e dirigidas pela equipe de qualidade da ODS, enquanto que as validações dos artefatos de software foram realizadas por uma fábrica de teste contratadas diretamente pela empresa cliente da ODS.

Com isso, uma única lista de revisão era usada tanto para a verificação quanto para a validação, conforme abaixo indicado:

- Modelo do Negócio (verificação e validação).
- Documento de Visão (verificação e validação).
- Modelo de Casos de Uso (verificação e validação).
- Modelo de Interface do Usuário (verificação e validação).
- Modelo de *Design* (verificação).
- Modelo de Dados (verificação).
- Modelo de Implementação (verificação).
- Modelo de Implantação (verificação).

Cada artefato acima (produto intermediário) possuía sua lista de revisão. A aplicação da lista de revisão na verificação/validação de um produto intermediário produzia um relatório de revisão (Anexo C). Ou seja, a aplicação de uma lista de revisão em um artefato tratado em uma iteração correspondia a um relatório de revisão. Assim, um relatório de revisão é uma instância de uma lista de revisão aplicada em uma iteração. Sempre que um artefato era criado, havia um relatório de revisão para a sua verificação/validação. Da mesma forma, para cada iteração que fizesse uma atualização daquele artefato, havia um relatório de revisão a ser utilizado na respectiva verificação/validação da atualização.

Em cada iteração era aplicado um conjunto de listas de revisão (com seu respectivo relatório de revisão), uma para cada artefato tratado na iteração em foco.

As validações foram realizadas pelos usuários com o suporte da equipe de qualidade da ODS.

Na primeira iteração, por ocasião das verificações, foi aplicada a técnica de revisão por pares (Laintenberger, at al. 2002; apud [SOF2006]) usando

profissionais de outros departamentos, porém isso apresentou problemas porque os itens de revisão requerem conhecimentos que, nem sempre, um profissional de outro departamento possui. Por exemplo, é mais fácil ensinar UML a um projetista de um outro projeto que não conhece UML, mas sabe OO, princípios de arquitetura e padrões de projeto, do que ensinar o negócio a ele. De outra forma, é mais fácil ensinar modelagem de casos de uso a um outro analisa de sistema que não conheça UML, mas conheça o negócio, do que ensinar negócio a um projetista ou a um analista-programador. Devido a essa dificuldade, decidiu-se que as revisões por pares seriam feitas por pares do mesmo departamento, mas que não tenham participado da criação ou modificação do artefato em questão.

É válido salientar que as revisões foram feitas diretamente no arquivo (.EAP) produzido pela ferramenta de modelagem Enterprise Architect (EA) usada no projeto. O EA oferece todos os recursos necessários para uma completa e detalhada descrição dos atores e seus casos de uso com fluxos de eventos e demais elementos de descrição; classes com seus respectivos atributos e operações; pacotes, atividades, estados, componentes e nós. Esses recursos foram úteis na agilização de todo o processo de desenvolvimento porque evitou a necessidade de elaboração de documentação adicional na forma de textos com uso do MS-Word.

Para cada item de revisão que apresentava erro, era anotada a quantidade de erros identificados no referido item e feita uma observação em um campo específico no formulário utilizado. Essa observação era um *feedback* fornecido pelo revisor a ser usado pelo desenvolvedor para a devida correção. Sempre que um produto intermediário (pertencente à lista acima descrita) fosse criado deveria se submeter a uma verificação e posterior validação. Sempre que esse mesmo produto sofresse qualquer tipo de atualização (correção ou evolução), também se submetia a uma nova verificação e validação. É por isso que

produtos intermediários sofriam verificações e validações em várias iterações de mais de uma fase.

Após a conclusão da revisão, o revisor usava o processo de configuração e mudança para armazenar no CVS a lista de revisão utilizada e comunicava o resultado da verificação aos envolvidos através de e-mail. Esse processo se repetiu até que nenhum erro fosse identificado. A quantidade máxima de revisões feitas para zerar os erros foi de três verificações/validações.

Enquanto que uma lista de revisão é usada em um produto intermediário que é criado em uma iteração e atualizada em várias outras, os casos de teste abordam aspectos relacionados à qualidade de um produto de software criado em uma única iteração. Proporciona teste da realização dos requisitos funcionais e não funcionais.

Com isso um conjunto de casos de teste formava um plano de teste para ser aplicado em um *build* que era a implementação dos casos de uso realizados na iteração em questão.

A criação dos casos de teste foi feita nas seguintes ocasiões:

- Modelagem do negócio.
- Modelagem dos casos de uso.
- Modelagem das interfaces gráficas dos usuários.
- Prototipação das interfaces gráficas dos usuários.
- Definição dos requisitos não-funcionais.
- Definição da arquitetura.
- Definição dos processos e *threads*[19] simultâneos.
- Modelagem de dados.
- Na modelagem de *design.*

[19] *Thread* são frações atômicas e indivisíveis de um processo executado por um software.

- Na modelagem de implantação.
- Na modelagem de implementação.

A estratégia de testes aplicados em cada iteração foi uma adaptação do que foi sugerido por [PRE2002], [SOF2006] e [YOU1990]. Em cada iteração houve os seguintes testes:

- Testes unitários realizados pela FS2. Os resultados desses testes não foram medidos pela ODS.
- Testes de integração efetuados pela ODS com o objetivo de verificar o funcionamento do *build* construído na iteração em foco e integrado aos demais *builds* já construídos e integrados em iterações anteriores. Estes testes foram medidos pela ODS.
- Teste de sistema, realizado por uma fábrica de teste contratada pela cliente da ODS. Estes testes não foram medidos.

Para cada iteração foi preparado um plano de teste contendo casos de teste em função dos casos de uso que foram realizados na iteração em foco. Esse plano de teste era aplicado internamente pela equipe do departamento de Gestão de Requisitos produzindo um relatório de testes bastante semelhante ao relatório de revisão. A única diferença era que os itens de revisão eram substituídos por casos de teste.

Todos os relatórios de revisão e os relatórios de testes foram sumarizados para a produção dos dados estatísticos que formaram a base de dados para a comprovação das hipóteses propostas pelo presente trabalho.

Em resumo, o processo operacional de verificação/validação é formado pelos seguintes passos:

1. Gerente de qualidade recebe o e-mail comunicando que existe uma SS ou uma OSI para ser verificação/validação.

2. O gerente de qualidade designa um analista de qualidade para conduzir o processo de verificação/validação.
3. O gerente de qualidade convoca os revisores ou os usuários validadores.
4. O analista de qualidade faz o *checkout*[20] da SS/OSI e dos artefatos a serem verificados/validados.
5. A equipe de verificadores/validadores aplica as listas de revisão (para produtos intermediários) ou plano de teste (para produtos de software).
6. O analista de qualidade registra os dados da verificação/validação no relatório de revisão ou relatório de teste.
7. O analista de qualidade faz o *checkin* da SS/OSI juntamente com os relatórios produzidos na verificação/validação.
8. O analista de qualidade passa e-mail para o gerente de qualidade com cópia para o gerente do projeto e o departamento ou FS envolvidos na produção, informando o resultado na verificação/validação.

6.5.3 Aplicação do Processo Operacional

A presente seção irá descrever as atividades do processo operacional, conforme apresentado na figura 12, o qual descreve o formato inicial, antes da aplicação da pesquisa-ação. Ao longo do texto a seguir serão mostrados os pontos do processo que sofreram melhoramentos, os problemas ocorridos, a definição inicial do processo e o seu formato final após o melhoramento.

A gestão das interações entre a ODS e as duas FSs envolvidas no projeto se revelou bastante árdua, dinâmica e cheia de riscos que, se não resolvidos, poderão elevar o grau de complexidade da condução do projeto de tal forma que poderá inviabilizá-lo. Isso justifica a utilização da pesquisa-ação como modalidade de pesquisa experimental, porque permitiu a interferência no processo de pesquisa em busca de ajustes necessários para que o processo fosse refinado mitigando todos os riscos acima referidos.

---

[20] *Checkout* é a extração de um arquivo da ferramenta de gerenciamento de configuração.

O projeto escolhido para a pesquisa-ação tinha uma característica que deu uma grande contribuição para ao sucesso do projeto, a qual foi o fato do cliente se situar no Rio de Janeiro e a ODS em São Paulo. Isso permitiu a elaboração de um cenário-padrão para todas as iterações constituído por quatro escopos:

- ODS executando o escopo **NR** (Modelagem do Negócio e Requisitos) no Rio de Janeiro.
- FS1 executando o escopo **D** (Análise-*Design*) em São Paulo.
- FS2 executando o escopo **P** (Implementação) em São Paulo.
- ODS executando o escopo **TI** (Teste e Implantação) em São Paulo.

A Figura 16 apresenta a visão geral do RUP com os cenários e seus respectivos escopos. É válido notar que em cada cenário a ODS executou um escopo duplo (NR e TI). Executando o escopo NR, a ODS produziu os insumos para a primeira FS, representada na Figura 16 como FS1, a qual executou somente a disciplina de Análise-*Design*. Não houve comunicação entre as duas equipes. A FS1 produziu um Modelo de *Design* o qual foi usado como insumo para uma segunda FS, representado na Figura 16 como FS2, executando a disciplina de Implementação para produzir o Modelo de Implementação e os artefatos de software testados unitariamente. Também não houve comunicação entre as duas equipes. As demais disciplinas (teste, implantação, gestão de configuração e mudança, gestão de projetos e ambiente) foram executadas pela ODS. Com isso passamos a ter uma parte da ODS no Rio de Janeiro, uma segunda parte da ODS em São Paulo juntamente com as duas fábricas (FS1 e FS2).

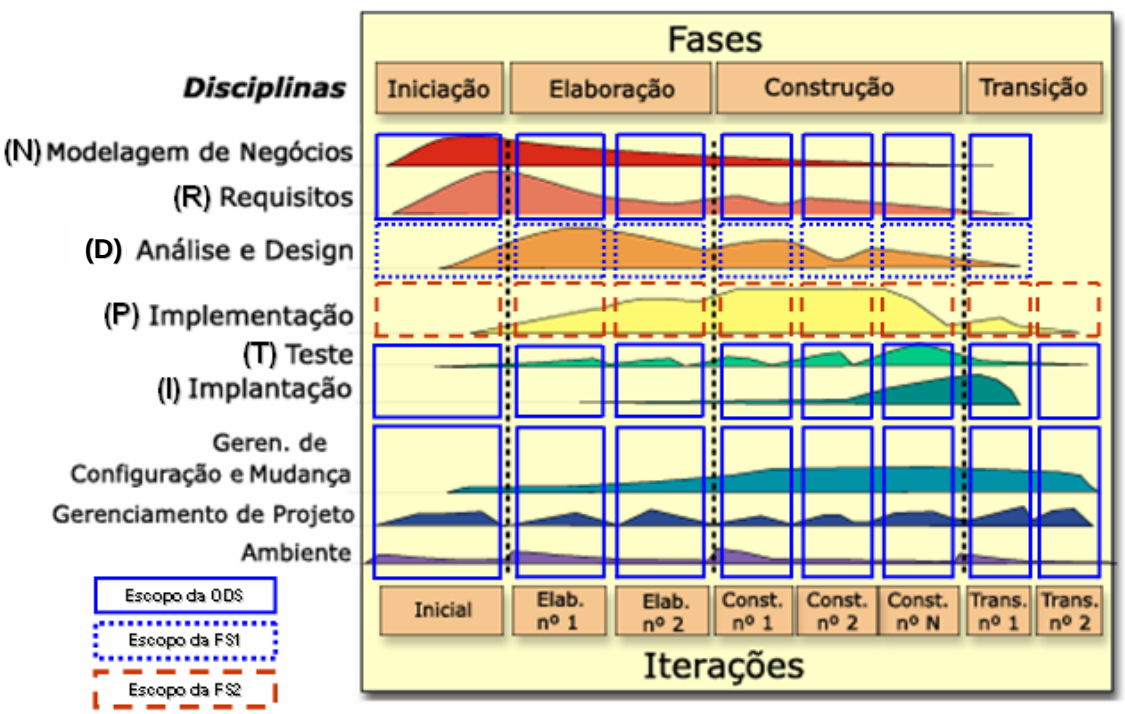


Figura 16 – Estratégia de Cenários e Escopos da Pesquisa-Ação (adaptação do RUP [RUP2002]).

Visando a garantir a ausência de reuniões para complementação dos requisitos, foi estabelecida uma norma interna que toda e qualquer documentação passada para a FS deveria ser por meio de uma SS usando o processo de gestão de configuração e mudança.

O processo operacional e o processo configuração e mudança tiveram papeis fundamentais em busca do estanqueamento dessas ambientes (FS1, FS2, ODS-Rio e ODS-SP) simulando uma separação geográfica entre as organizações.

A primeira atividade do processo operacional (figura 12) proposto no Capítulo 5 foi o “Planejamento de Cenários e Escopos”. Esta atividade sempre precede a execução de qualquer escopo executado pela ODS ou por uma FS. Devido ao fato de todas as iterações de todas as fases possuírem o mesmo cenário, ou

seja, os mesmos escopos para a ODS e para as duas FS, foram elaboradas as seguintes OSIs:

- OSI para o Departamento de Gestão de Requisitos executar o escopo **NR**.
- OSI para o Departamento de Gestão da Qualidade executar a verificação e validação dos artefatos produzidos pelo departamento de requisitos.
- OSI para o Departamento de Gestão da Qualidade executar a verificação dos artefatos originários da FS1.
- OSI para o Departamento de Gestão de Qualidade executar as verificações e validações dos testes dos componentes produzidos pela FS2.
- OSI para o Departamento de Gestão de Configuração e Mudança para criar *baseline* e fazer a implantação do módulo desenvolvido na iteração.

Seguindo o processo operacional, no seu formato inicial descrito na figura 12, sempre que qualquer departamento interno da ODS necessitar fazer alguma ação no projeto, será necessária a interferência do gerente do projeto elaborando uma OSI autorizando sua execução. Logo no início do projeto, percebeu-se que isso gerava dois problemas:

1. Um problema burocrático, pois nem sempre o gerente do projeto estava disponível para a elaboração da OSI e por isso o processo parava causando impacto no cumprimento dos prazos.
2. Dispersão dos registros dos eventos ocorridos ao longo do ciclo de vida de um artefato. Conforme demonstrado na Figura 7, uma OSI é um documento de mudança aplicado em um único escopo de uma iteração a ser executado por um departamento da ODS. Com isso, o Modelo de Casos de Uso, por exemplo, desenvolvido na ODS tem seus eventos de elaboração registrados em uma OSI e os eventos referentes ao escopo de Teste e Implantação (TI) registrados em mais duas outras OSIs (uma para o Departamento de Gestão da Qualidade e outra para o Departamento de Implantação).

Para resolver o problema burocrático, buscou-se a redução da interferência operacional do gerente na execução do processo. A solução encontrada foi fazer com que o processo fluísse automaticamente sem a interferência do gerente do projeto. O processo ficaria de tal forma que, sempre que um escopo de produção[21] fosse concluído (internamente ou por uma FS), o próximo escopo fosse o de Teste e Implantação (TI). A passagem do controle do fluxo seria feita por meio de um e-mail previsto no processo de configuração e mudança. Esse e-mail era endereçado à unidade que iria executar a próxima etapa do processo, com cópia para o gerente do projeto. A unidade executora do trabalho saberia qual o próximo escopo a ser executado e qual unidade seria utilizada. Com isso, o processo passou a fluir sem a interferência ativa do gerente do projeto, o qual fazia somente a supervisão do fluxo de trabalho.

Esta solução apresentou vantagens e desvantagens. As vantagens residem no fato de realmente liberar o gerente das atividades operacionais, dando celeridade ao processo. Contudo, depois de duas iterações, percebeu-se um pouco de displicência nas equipes dos diversos departamentos da OSI. O motivo dessa displicência foi a falta de um controle mais rígido, que antes era feito pelo gerente do projeto, passando a impressão de que o projeto estava abandonado. Além da perda de produtividade, houve também uma queda na qualidade dos artefatos produzidos, porque as falhas encontradas mostravam que estavam ocorrendo descuidos por parte das equipes produtoras, característica de quem não está comprometido com a qualidade. Infelizmente, não foi possível fazer uma medição para precisar a influência da presença do gerente na qualidade dos artefatos produzidos nos projetos de software.

---

[21] Escopo de Produção é um escopo pertencente a um cenário, conforme definido no capítulo 5 deste trabalho, que executa, pelo menos, uma das seguintes disciplinas: Modelagem do Negócio, Requisitos, Análise-*Design* e Implementação.

Percebeu-se, ainda, que, em um ambiente de desenvolvimento de múltiplos projetos, no qual uma equipe atua em vários projetos em diferentes fases do processo produtivo, esse tipo de automaticidade do processo não funciona adequadamente, porque as equipes não têm como definir a prioridade de cada OSI que entra em execução simultaneamente. Por isso, ocorreu uma disputa por recursos entre os gerentes de projetos, causando desgastes desnecessários.

Para solucionar o problema, foi criado um novo departamento da ODS, chamado de “Escritório de Engenharia de Software”, com o objetivo de centralizar o seguinte:

- Controle dos fluxos operacionais (mecanismo proposto por este trabalho) dos múltiplos projetos que são desenvolvidos simultaneamente.
- Controle dos ciclos de vida de cada escopo executado pela ODS.
- Controle das interações com as FSs envolvidas no processo produtivo.
- Controle da utilização dos recursos humanos, infra-estrutura disponível e FSs subcontratadas, buscando a maximização de suas utilizações nos múltiplos projetos de software que são desenvolvidos simultaneamente.
- Suporte ao Escritório de Projetos no planejamento de cenários e escopos da ODS e das FS envolvidas em cada projeto de desenvolvimento de software.
- Definição, em conjunto com o Escritório de Projetos, das prioridades de cada OSI em função dos cronogramas de entregas de produtos de software aos clientes contratantes.

É válido notar que o Escritório de Engenharia de Software não gerencia o projeto. Sua unidade de gerenciamento é a iteração como seus respectivos escopos executados internamente na ODS ou por FSs. Continua sob a responsabilidade do Escritório de Projetos todo o planejamento do projeto, a distribuição dos casos de uso nas iterações e a estratégia de cenários e escopos usados no projeto.

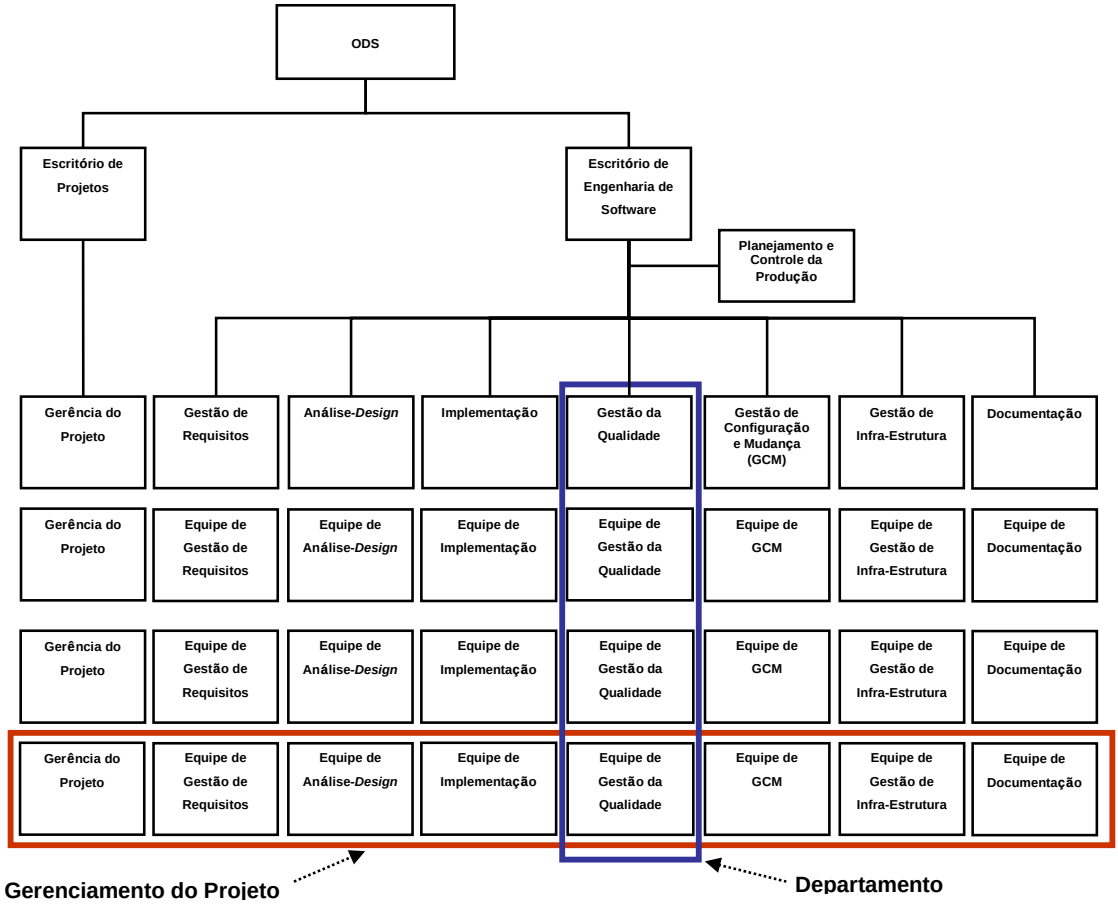


Figura 17 – Organograma da ODS após a inclusão do Escritório de Engenharia de Software

No presente projeto, o Escritório de Engenharia de Software ficou no organograma (figura 17) hierarquicamente posicionado em paralelo ao Escritório de Projetos, ambos reportando-se ao CTO (*Chief Technology Office*) da empresa. O gerente do projeto fica livre das atividades operacionais do projeto, as equipes não ficam “soltas” e as prioridades são definidas em conjunto com os demais gerentes dos diversos projetos em desenvolvimento.

No que diz respeito à dispersão dos registros dos eventos ocorridos ao longo do ciclo de vida de um artefato, a solução encontrada foi a alteração do conceito da OSI e da SS. Antes, esses documentos estavam associados a um único escopo que é executado em um departamento (no caso de uma OSI) ou uma FS (no

caso de uma SS). Com essa mudança de conceito, a OSI e a SS ficaram associadas ao ciclo de vida de um escopo de produção.

A Figura 18 mostra como ficou o modelo do cenário (após as adaptações acima referidas) originalmente apresentado na figura 7. Se um escopo de produção for executado pela ODS o documento de mudança utilizado será a OSI. É válido notar que esta OSI acompanhará todo o ciclo de vida do escopo de produção entre os departamentos da ODS que participarem do processo produtivo. No presente projeto, este foi o caso de escopo **NR** com o ciclo de vida totalmente executado dentro da ODS.

Porém, se um escopo de produção que já teve seu processo produtivo iniciado na ODS (já possui uma OSI) tiver a necessidade de ter parte do seu ciclo de vida executado complementarmente em uma FS, será necessária a criação de uma SS para aquela FS envolvida no processo produtivo. Neste caso, o escopo de produção terá uma única OSI e uma SS para cada FS utilizada no seu ciclo de vida do escopo de produção em questão. Isso pode acontecer em casos de utilização de fábrica de testes usada para validar os produtos intermediários e os artefatos de software produzidos internamente pela ODS. Não foi o caso do presente projeto.

Se um escopo de produção for executado por uma FS, o documento de mudança será a SS, a qual acompanhará todo o ciclo de vida do escopo dentro da FS em questão. Se o ciclo de vida deste escopo tiver parte executada dentro da ODS, não será necessária a criação de uma OSI para acompanhar os trabalhos realizados pelos departamentos da ODS. Neste caso, a mesma SS continuará sendo usada para registrar todos os eventos ocorridos na ODS. Este caso ocorreu no escopo **D** (da FS1) e no escopo **P** (da FS2).

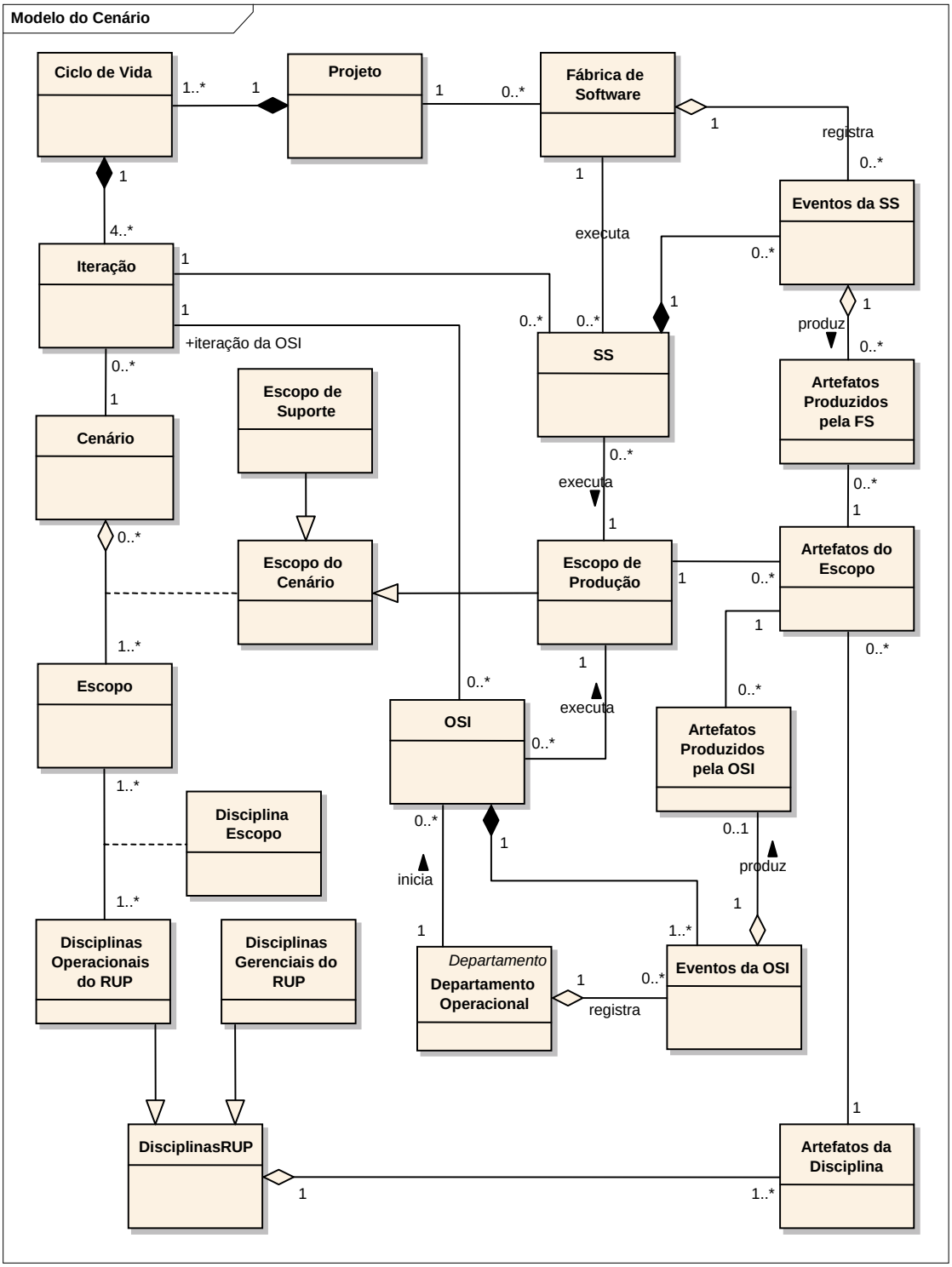


Figura 18 – Modelo do cenário – modificado pela pesquisa-ação

Contudo, se este mesmo escopo tiver parte do seu ciclo de vida executado por uma segunda FS, será necessária a criação de uma nova SS para essa segunda FS envolvida no processo produtivo. Este caso também não ocorreu no presente projeto.

É válido lembrar que estas decisões relacionadas à criação de OSIs e SSs, acima descritas, referem-se ao ciclo de vida de escopo de produção. É necessário ter cuidado para não confundir com o ciclo de vida de uma iteração. Por exemplo, as disciplinas de Análise-*Design* (escopo de produção **D**) e Implementação (escopo de produção **P**) são dois escopos de produção distintos, cada qual com seu respectivo ciclo de vida. O escopo **D,** com uma SS para a FS1 e o escopo **P,** com a SS para a FS2. Embora sejam dois escopos de produção, é válido lembrar que o produto do escopo de produção **D** (da FS1) é insumo para o escopo de produção **P** (da FS2). Isso não quer dizer que essas duas disciplinas pertençam ao mesmo escopo de produção ou tenha um único ciclo de vida.

Nota-se, ainda, que ambos os escopos de produção (**D** e **P**) solicitam a colaboração do escopo TI (Teste e Implantação) que é executado respectivamente pelo Departamento de Gestão de Qualidade e pelo Departamento de Implantação.

A definição das disciplinas que farão parte de um escopo é feita pela atividade “Planejamento de Cenários e Escopos”, que é a primeira atividade do processo operacional (Figuras 12 e 19).

Para simplificar, pensou-se em unificar a OSI e a SS em um único documento de mudança, mas isso não se mostrou interessante porque uma OSI iniciada na ODS quando fosse enviada para uma FS levaria consigo um histórico que não interessaria àquela FS envolvida no processo. Por outro lado, uma SS iniciada

em uma FS e com parte do seu ciclo de vida executado na ODS, mostrou-se bastante interessante por causa dos registros dos eventos ocorridos na FS.

A Figura 19 mostra o estado final do processo operacional após as modificações ocorridas na pesquisa-ação. As atividades foram distribuídas em duas raias: "Planejamento e Controle da Produção" e "Destinatários (ODS/FS)".

"Planejamento e Controle da Produção" é uma célula de trabalho dentro do Escritório de Projetos que passou a controlar o processo através das atividades constantes na raia. Os Destinatários são os departamentos da ODS (departamento de produção e departamento de suporte à produção) e as FSs envolvidas no processo produtivo.

É válido notar que o processo operacional, embora pareça longo, é simples. Aparenta ser longo porque apresenta aspectos do processo de configuração e mudança, mostrando os *checkin, checkout,* além da emissão de e-mails como forma de comunicação entre as unidades de produção da ODS e da FS com a célula de Planejamento e Controle da Produção.

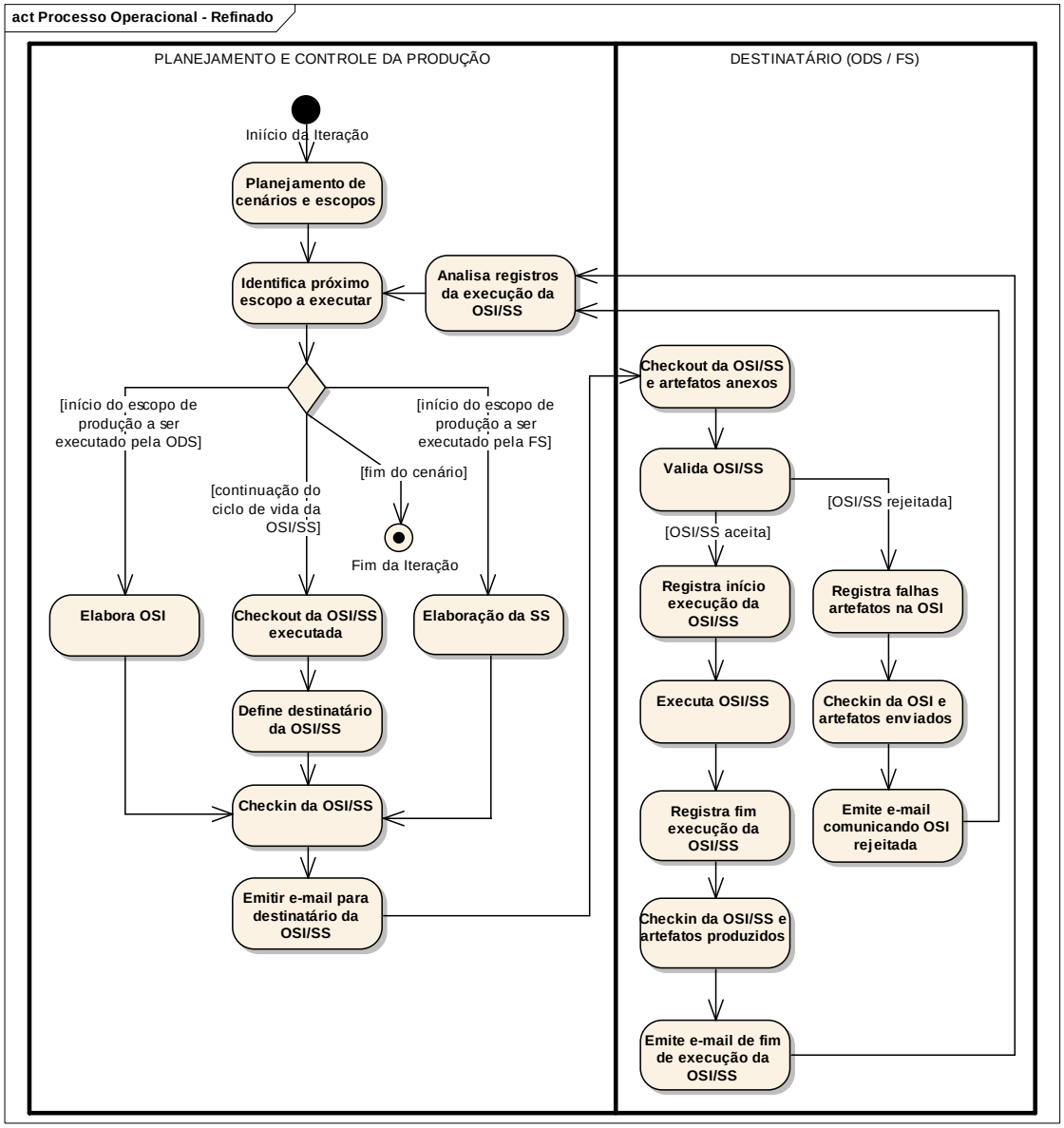


Figura 19 – Processo operacional refinado (após modificações promovidos na pesquisa-ação)

6.6 Normas Internas

Para que o projeto fosse bem sucedido foi criada uma série de normas internas à ODS. Essas normas fazem parte dos mecanismos objeto da presente pesquisa-ação. O processo de definição dessas normas foi incremental. Na medida em que se fazia necessária, uma nova norma era estabelecida ou refinada e passava a constar de um dos guias abaixo listados:

- Guia de Modelagem de Casos de Uso.
- Guia de Análise-*Design*.
- Guia de Implementação.

6.6.1 Guia de Modelagem de Casos de Uso

- Regras de nomenclatura de casos de uso.
- Recomendações para a descrição sucinta de casos de uso.
- Recomendações para a descrição de *includes* dentro dos fluxos de eventos dos casos de uso.
- Recomendações para a descrição de pontos de extensão na descrição do caso de uso base.
- Recomendações para a descrição de generalizações de casos de uso.
- Recomendações para a descrição de casos de uso abstratos que foram fatorados dos casos de uso base que usaram associação *include, extend* ou generalização.
- Recomendações para descrição de fluxo principal, alternativos e de exceção.
- Recomendações para descrição de requisitos não-funcionais.
- Recomendações para descrição de regras de negócio.
- Recomendações para a descrição do Glossário.
- Recomendações para a criação das Mensagens do Sistema.

6.6.2 Guia de Análise-*Design*

- Regras de nomenclatura de Classes de Fronteira, Classes de Serviço (Ativas), Classes de Entidade, Classes Abstratas, Interfaces, Atributos, Operações, Relacionamentos, Papeis, Mensagens, Subsistemas, Pacotes, Diagramas.
- Regras de utilização de *Design Pattern*.
- Recomendações para descrição de Classes, Atributos, Operações, Subsistemas, Pacotes, Componentes e Nós.

- Recomendações para a definição das visões arquiteturais de casos de uso, projeto, processo e implantação.
- Recomendações para a definição de subsistemas, camadas arquiteturais, pastas, para o modelo de *design*.
- Recomendações para a realização de casos de uso (diagramas de interação).
- Recomendações para a elaboração de diagramas de classes de projeto, diagramas de máquina de estado.
- Recomendação na elaboração do Modelo de Implantação.

6.6.3 Guia de Implementação

- Regras de nomenclatura de Componentes.
- Regras para o mapeamento entre classes de projeto e classes de implementação.
- Regras para o mapeamento entre subsistemas e pacotes de projeto para respectivamente componentes e pacotes de implementação.
- Recomendações para a definição das visões arquiteturais de implementação.
- Recomendações para a definição de rastreabilidade desde a necessidade até o componente.
- Recomendações para a elaboração de diagramas de classes de implementação.

6.7 Comprovação das Hipóteses

A presente seção irá comprovar as hipóteses apresentadas no Capítulo 1 associando-as aos mecanismos definidos no Capítulo 5.

6.7.1 Comprovação da Hipótese-1

Enunciado da Hipótese-1:

Os pontos, dentro do fluxo de uma iteração, nos quais a ODS deverá acionar a FS dependem do cenário estabelecido para a iteração em foco, onde cada uma dessas organizações deverá ter um escopo de atuação executando atividades pertencentes às disciplinas operacionais do RUP.

Os mecanismos associados à hipótese-1 são:

- O conjunto de escopos constituídos por uma ou mais disciplinas do RUP.
- Os cenários, em que cada um é formando por um arranjo de escopos de tal forma que a soma de todos os escopos de um cenário cubra todas as disciplinas operacionais do RUP.

A comprovação da hipótese-1 se dá na ocasião do planejamento de um cenário composto por um arranjo de escopos. Os pontos principais desse planejamento estão nas seguintes definições:

- Quais os cenários e respectivos escopos que deverão ser usados nas iterações de cada fase do processo de desenvolvimento (Iniciação, Elaboração, Construção e Transição).
- Quem irá executar cada escopo de cada cenário de cada iteração de cada fase.

Cada um dos escopos de cada cenário definido deverá ser executado de forma integrada entre a ODS e uma ou mais FSs. Conforme demonstrado nas Figuras 8 e 9, os escopos podem variar entre os cenários. É possível que haja cenários que a ODS execute um escopo e subcontrate uma ou mais FSs para executar outros escopos. Também é possível que haja cenários que a ODS execute nenhum escopo, se restringindo exclusivamente a controlar todo o processo de interação com as FSs envolvidas na execução dos escopos do cenário.

Com isso, devido ao fato de um escopo ser formado por uma ou mais disciplinas do RUP, o ponto de acionamento de uma FS é a disciplina que inicia o escopo

destinado à FS subcontratada, independentemente se a ODS irá ou não executar o escopo imediatamente anterior ao escopo subcontratado.

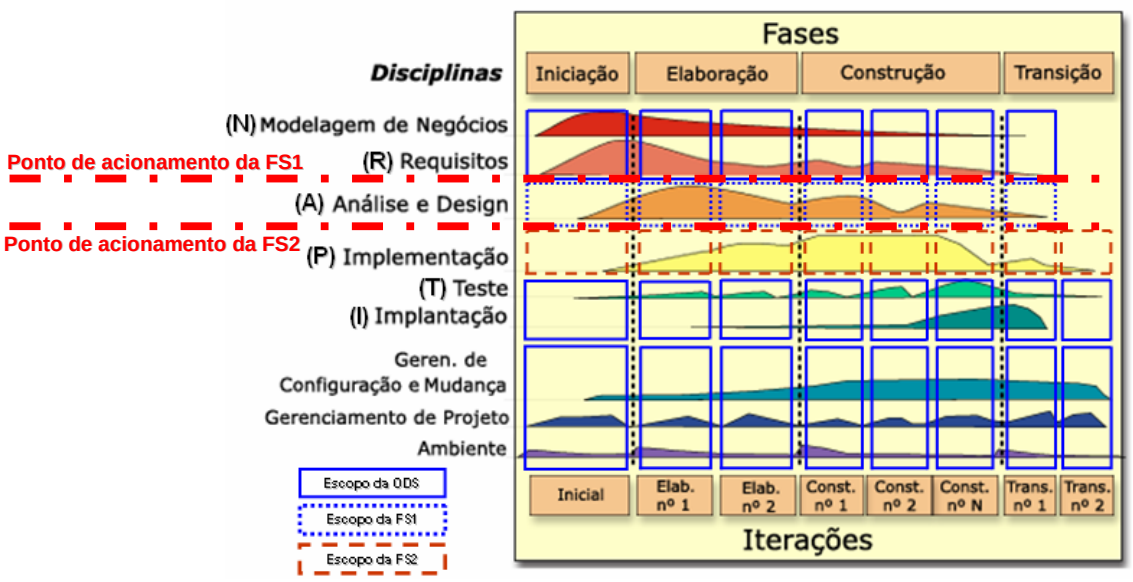


Figura 20 – Pontos de acionamento de duas FSs

A figura 20 mostra os dois pontos de acionamento das duas FSs usadas na pesquisa-ação. Contudo esses pontos poderiam variar em função do planejamento dos cenários. Do ponto de vista técnico, o gerente do projeto tem total liberdade para definir tantos quantos pontos de acionamento ele desejar. Porém do ponto de vista econômico, se faz necessário uma ponderação no sentido de manter uma constância entre os cenários para viabilizar as operações dentro da FS subcontratada.

6.7.2 Comprovação da Hipótese-2

Enunciado da Hipótese-2:

> *Os artefatos que a ODS deverá produzir em uma iteração para serem enviados para a FS dependem do escopo de atuação da ODS e da(s) FSs envolvidas na iteração em foco.*

Cada disciplina do RUP se constitui em uma unidade atômica de formação de um escopo, ou seja, o menor tamanho de um escopo é uma disciplina operacional do RUP. A conseqüência disso é que o produto de um escopo deverá ser o conjunto de artefatos obtidos na execução de cada disciplina envolvida no escopo. Por exemplo: se o escopo da ODS for formado pelas disciplinas NR (Modelagem do Negócio e Requisitos), conforme mostrado na tabela 1 e na figura 16, os artefatos que deverão ser produzidos são aqueles correspondentes aos produtos das disciplinas de Modelagem do Negócio e Requisitos, conforme demonstrado na tabela 2.

Conforme demonstrado na seção 6.7.1 que trata da comprovação da hipótese-1, a ODS poderá, ou não, executar algum escopo. Logo, os artefatos que a ODS irá produzir dependerão disso. Na hipótese da ODS executar algum escopo, ela irá gerar os artefatos produtos das disciplinas envolvidas nesse escopo. Esses artefatos irão compor uma SS que será endereçada para alguma FS que irá executar o escopo imediatamente posterior (no fluxo da iteração) ao escopo executado pela ODS.

6.7.3 Comprovação da Hipótese-3

Enunciado da Hipótese-3:

> *Os artefatos necessários para que uma FS possa gerar o produto solicitado em uma iteração e o faça de forma desacoplada da ODS, depende do escopo da FS na iteração em foco, da correta utilização da UML na especificação desses artefatos e do pleno conhecimento em UML por parte dos profissionais da FS envolvidos no desenvolvimento dos produtos solicitados.*

A comprovação dessa hipótese requer uma abordagem das seguintes dependências:

- Dependência do escopo da FS subcontratada para a definição dos artefatos que ela irá necessitar.

- Dependência da correta utilização da UML na especificação dos artefatos que serão usados como insumos para a FS subcontratada.
- Dependência do pleno conhecimento da UML por parte da FS subcontratada para um completo entendimento das especificações recebidas e, a partir daí, produzir os artefatos indicados pelo escopo usado pela FS subcontratada sem a necessidade de qualquer tipo de comunicação com a organização produtora dos insumos.

O alvo a ser alcançado nessa comprovação é mostrar que uma FS subcontratada pode atuar iterativamente com a ODS de forma totalmente desacoplada. Os pontos focais são:

- Suficiência do conjunto de artefatos insumos.
- UML como instrumento de comunicação.

O objetivo da presente pesquisa-ação é mostrar que uma FS é capaz de produzir, por repetidas vezes, um conjunto de artefatos (saída) caracterizados como produtos de um escopo previamente planejado, a partir de um conjunto de artefatos (insumos) especificados através da UML, resultante da execução de um escopo anterior dentro do fluxo de uma iteração, sob a condição de não haver qualquer tipo de comunicação entre a organização produtora dos insumos e a FS subcontratada.

6.7.4 Comprovação da Hipótese-4

Enunciado da Hipótese 4:

Para que a ODS possa controlar as interações com uma ou mais FSs envolvidas no cenário de cada iteração, é necessário um processo que execute o planejamento dos cenários e respectivos escopos da ODS e de cada FS envolvida na iteração e controle a execução das disciplinas operacionais do RUP dentro dos escopos da ODS e das FS envolvidas.

Para controlar as interações ODS x FS se faz necessário um processo com os seguintes requisitos:

- Permitir o uso de qualquer tipo de cenário com qualquer tipo de configuração de escopos, sem alterar o processo interno das disciplinas do RUP.
- Permitir mudança de cenários ao longo de um ciclo de desenvolvimento ou de evolução.
- Permitir utilização de mais de uma FS em cada cenário.
- Permitir a ODS executar nenhum, um ou mais escopos.
- Permitir que uma FS possa executar mais de um escopo em uma iteração.
- Permitir a substituição de uma FS no meio de um ciclo de desenvolvimento ou de evolução.

A presente pesquisa-ação desenvolveu um processo operacional, conforme demonstrado na figura 12. Este processo foi modificado várias vezes em busca do atendimento dos requisitos acima descritos.

A comprovação da hipótese-4 irá ocorrer se o processo conduzir adequadamente todas as interações ODS x FS e controlar todas as atividades executadas pelos diversos departamentos da ODS.

6.8 Resultados Obtidos

O objetivo desta seção é mostrar os dados obtidos ao longo de toda a pesquisa-ação. A comprovação final das quatro hipóteses propostas pelo presente trabalho será feita através da medição dos erros cometidos pelas FSs envolvidas no projeto. Essa comprovação se dará se:

- A FS1 produzir o Modelo de *Design* devidamente verificado pelo departamento de qualidade, a partir do Modelo de Negócio, Modelo de Casos de Uso e do Guia de Análise-*Design*.

- A FS2 produzir o Modelo de Implementação devidamente verificado pelo departamento de qualidade, a partir do Modelo de Casos de Uso, Modelo de *Design,* Guia de Análise-*Design* e do Guia de Programação.
- Todas *builds* construídas forem corretamente integradas, verificadas pela equipe interna da ODS e validadas pelos usuários.

Os resultados foram obtidos através do processo de qualidade, o qual produziu relatórios de verificação de produtos intermediários e relatórios de teste de software. Esses relatórios foram compilados e produziram os dados que serão apresentados na presente seção e nos Anexos C e D. O Anexo C contém todos as listas de verificações que foram utilizadas pela equipe de qualidade na validação dos artefatos produzidos internamente pela ODS e pelas duas FSs. O Anexo D contém os dados que foram compilados a partir dos das listas de verificações devidamente preenchidas por ocasião das validações por pares.

6.8.1 Verificação e validação dos Produtos Intermediários

Foram feitas revisões em oito produtos intermediários abaixo listados os quais foram produzidos pela ODS e pelas FS1 e FS2:

Produtos intermediários originários da ODS:

1. Modelo do Negócio.
2. Documento de Visão do Sistema.
3. Modelo de Casos de Uso.
4. Requisitos.
5. Modelo de Interface do Usuário.

Produtos intermediários originários da FS1:

6. Modelo do Projeto.
7. Modelo de Dados.
8. Modelo de Implantação.

Produto intermediário originário da FS2:

9. Modelo de Implementação.

Para tal, o processo de qualidade utilizou listas de revisão que foram aplicadas dentro do ciclo de vida do escopo que produziu cada artefato. Cada artefato possuía sua lista de revisão a qual foi desenvolvida pela ODS e homologada pelos *stakeholders.* Esse processo de verificação foi chamado de Revisão por Pares, porque os revisores eram profissionais com o mesmo perfil dos que elaboraram o artefato em revisão.

As revisões desses artefatos envolveram um conjunto de itens de revisão que abordaram aspectos subjetivos (veja listas de revisão no Anexo C), relacionados ao negócio. Por isso os usuários também eram envolvidos na revisão. Esse envolvimento dos usuários trouxe um problema de agendamento, porque os executivos envolvidos no projeto nem sempre tinham tempo nem condições de leitura dos artefatos produzidos por desconhecerem a UML. Além disso os usuários ficavam no Rio de Janeiro e a ODS em São Paulo.

Para resolver esse problema, ficou decidido que a revisão dos artefatos intermediários pelos usuários deveria ser feita através da técnica de *walkthrough* combinada com inspeções [FIO1998], que consistiu de uma apresentação, onde um membro da equipe produtora apresentava o artefato a ser revisado, após o que a lista de revisão era utilizada para dirigir a inspeção, ocasião em que os usuários apontavam os erros os quais eram descritos na lista de revisão do artefato em revisão.

Devido ao fato dos usuários envolvidos na revisão não terem conhecimento necessário para entender alguns aspectos do artefato foco da revisão, antes da aplicação das técnicas de *walkthrough* e inspeção os usuários receberam um pequeno treinamento *on-the-job* sobre os elementos da UML usados no artefato em questão.

As reações dos usuários que foram percebidas ao longo do processo de desenvolvimento foram:

- Plena aceitação da idéia do treinamento "*on-the-job*".
- Os usuários entenderam a estrutura e a sintaxe da UML.
- A fluência da leitura dos artefatos UML foi aumentando na medida em que as seções de validação ocorriam.
- Depois das primeiras seções de validação, foi testado o envio dos artefatos com alguns dias de antecedência, para que os usuários pudessem antecipar seus comentários. Percebeu-se uma grande pró-atividade por parte dos usuários, fato que não ocorria antes das seções de validação.
- O conhecimento da UML por parte dos usuários reduziu o tempo gasto nas seções de validação, bem como trouxe os usuários como co-autores do projeto do sistema. Isso facilitou de sobremaneira todo o processo de qualidade.

Um fator crítico de sucesso para um completo envolvimento dos usuários, foi a não utilização do termo " treinamento". Esse termo foi substituído pela idéia de uma palestra sobre a metodologia de trabalho que deveria ser utilizada. Anteriormente, o gerente do projeto procurou, em reuniões com os usuários, verificar se um treinamento seria bem vindo pelos usuários. A conclusão foi que eles não estariam dispostos gastar tempo com treinamento nesta área, porque isso não faz parte do cotidiano profissional deles. Por isso, a forma de abordagem com os usuários é um fator crítico de sucesso para a obtenção dos seus envolvimentos em um projeto de software. Bastou a substituição do termo "treinamento" por "apresentação da metodologia de trabalho" para se conseguir o envolvimento dos usuários.

6.8.2 Teste de Software

Os testes de software foram aplicados em todas as iterações das fases de Elaboração, Construção e Transição, procurando identificar erros no produto final de software segundo a abordagem de “qualidade em uso” [ISO9126]. Primeiramente foram realizados os testes de verificação pela equipe da ODS para que os erros fossem identificados e resolvidos, para somente depois serem feitos os testes de validação pelos usuários.

Durante as fases de Elaboração e Construção, os testes foram realizados no ambiente de teste da ODS, porém na fase de Transição, todos os testes foram realizados no ambiente de homologação da empresa contratante do serviço de desenvolvimento. Os testes realizados foram:

- Testes da Interface Gráfica do Usuário – *Graphical User Interface* (GUI), usando uma lista de revisão, abordando aspectos relacionados aos padrões gráficos definidos pela empresa contratante. Este tipo de teste foi realizado em todas as interações das fases de Elaboração, Construção e Transição.
- Testes funcionais aplicados para validar o comportamento do sistema quanto ao atendimento dos requisitos funcionais e não-funcionais. Os testes funcionais seguiram um grande conjunto de casos de teste que foram produzidos pela equipe de qualidade ao longo do processo de definição dos processos de negócio, casos de uso e requisitos não funcionais. Este tipo de teste foi realizado somente nas fases de Construção e Transição.

Os produtos recebidos da FS eram na forma de um *build*. Cada iteração produzia um conjunto de *builds* que eram integrados ao sistema formando um novo *release.* O objeto dos testes era o conjunto de *builds* produzidos na iteração. Cada *build* possuía seu respectivo plano de teste o qual era definido por ocasião da elaboração dos modelos de negócio, de caso de uso, de *design*, de dados, de implantação e de implementação. Os planos de testes dos *builds*

da iteração eram reunidos em um plano de teste da iteração, os quais eram efetivamente utilizados tanto nos testes internos na ODS quanto pelos testes de validação feitos pelos usuários.

Infelizmente os testes de validação dos usuários não puderam ser tabulados. A estatística apresentada no presente projeto é a obtida nos testes de verificação executados internamente na ODS. O problema ocorrido era que, o ritmo de teste de validação pelos usuários nem sempre era compatível com o ritmo do projeto. Percebia-se que algumas iterações não eram corretamente testadas por eles. Para solucionar o problema a empresa cliente da ODS contratou uma fábrica de testes para efetuar os testes de validação necessários, fato que impossibilitou o controle da forma que era feito internamente na ODS.

6.8.3 Indicadores de Qualidade

Todas as listas de revisão e casos de teste foram compiladas e geraram um conjunto de indicadores de importância para a comprovação das hipóteses definidas no presente projeto de pesquisa. A tabulação de dados foi feita abordando os seguintes indicadores:

Total de Erros

Erros cometidos nas verificações dos produtos intermediários em cada verificação, iteração e fase.

Total de Erros / Elemento de Referência

Total de erros dividido pela quantidade de elementos de referência (casos de uso de negócio, caso de uso de software, característica de software [ISO9126] ou nó-dispositivo).

Percentual de Itens Errados

É a quantidade de itens de revisão que tiveram erros em relação ao total de itens verificados por ocasião de uma verificação.

Percentual de Erros Objetivos

É a soma de erros ocorridos em itens objetivos em relação à quantidade total de erros identificados em uma verificação. Item objetivo é um item de revisão que aborda aspectos relacionados ao correto uso da UML, do RUP, de normas, padrões e recomendações definidas pela gerência do projeto. Possui caráter exato. Não requer interpretação. O resultado deverá ser certo ou errado.

Percentual de Erros Subjetivos

É a soma de erros ocorridos em itens subjetivos em relação à quantidade total de erros identificados em uma verificação. Item subjetivo é um item de revisão que aborda aspectos relacionados ao negócio, à correta interpretação e descrição dos requisitos, regras de negócio, especificações e conceitos relacionados ao domínio do negócio. Possui caráter subjetivo, susceptível a uma interpretação. Esses itens devem ser verificados por um especialista no negócio (revisão por pares) [FIO1998].

6.8.4 Análise dos Indicadores de Qualidade

6.8.4.1 Análise Panorâmica dos Indicadores de Qualidade dos Produtos Intermediários

A tabela “Estatística de erros identificados em cada revisão dos produtos intermediários” no Anexo D apresenta uma visão panorâmica da estatística de erros cometidos em cada revisão de cada produto intermediário em cada iteração de cada fase do processo usado no projeto.

Os erros identificados nas revisões dos oito produtos intermediários sofreram dois níveis de totalizações:

- Total de erros por fase.
- Total de erro por revisão na iteração.

Nestes dois níveis de totalizações percebe-se claramente uma forte tendência de queda nos erros cometidos ao longo do processo. Percebe-se, entretanto, que a fase de Iniciação teve, nos dois níveis de totalizações, uma quantidade menor de erros do que a fase de Elaboração. Isso se deve ao fato de que quatro artefatos (Modelo de Interface do Usuário, Modelo do Projeto, Modelo de Dados e Modelo de Implementação) não foram criados naquela fase. É válido notar que esses quatro artefatos foram responsáveis por 93,99% dos erros cometidos na primeira verificação da fase de Elaboração.

Portanto, se esses artefatos fossem criados na fase de Iniciação, as totalizações de erros desta fase seriam maiores, ultrapassando a quantidade de erros cometidos na Elaboração, acentuando ainda mais a tendência de decréscimos das taxas de erros cometidos ao longo do processo.

Verificou-se que a redução dos erros proporcionou uma redução no tempo gasto no processo de qualidade porque a quantidade de revisões foi sendo reduzida a cada iteração. A tabela 4 mostra esse fato:

Artefatos	**Inic.**	**Elaboração**		**Construção**					**Transição**	
	I-1	**E-1**	**E-2**	**C-1**	**C-2**	**C-3**	**C-4**	**C-5**	**T-1**	**T-2**
Modelo do Negócio	3	2	2	0	0	0	0	0	0	0
Documento de Visão do Sistema	3	0	0	0	0	0	0	0	0	0
Modelo de Casos de Uso	3	3	2	2	2	0	0	0	0	0
Modelo de Interface do Usuário	NA	3	2	2	2	0	0	0	0	0
Modelo do Projeto	NA	3	2	2	2	0	0	0	0	0
Modelo de Dados	NA	3	2	0	0	0	0	0	0	0
Modelo de Implantação	2	2	0	0	0	0	0	0	0	0
Modelo de Implementação	NA	3	2	2	2	2	0	0	0	0
Total de verificações	**11**	**19**	**12**	**8**	**8**	**2**	**0**	**0**	**0**	**0**

Legenda: NA: a revisão não foi aplicada.

Tabela 4 - Freqüência de revisões em cada iteração

A fase de Iniciação teve onze revisões, portanto menor que as dezenove verificações ocorridas na primeira iteração da Elaboração. Isso ocorreu porque

quatro artefatos não foram criados na Iniciação, fato já acima comentado. Os motivos dessa redução são:

- As equipes de produção dos artefatos foram adquirindo experiência mostrando que cada iteração é como se fosse um sistema que estivesse sendo concluído. No final de cada iteração, as equipes envolvidas passavam um sentimento de conclusão de uma etapa do trabalho, bem semelhante ao que se sente na finalização de um projeto.
- Familiarização com os padrões de qualidade estabelecidos.
- Refinamento por ocasião do término de cada iteração do Guia de Modelagem de Casos de Uso, Guia de Análise-*Design* e o Guia de Programação que eram compartilhados pelas equipes internas da ODS e das FSs envolvidas.

6.8.4.2 Análise dos Produtos da ODS

Os artefatos criados pela ODS foram:

- Modelo de negócio;
- Documento de Visão;
- Modelo de casos de uso;
- Requisitos;
- Modelo de interface com o usuário.

O gráfico 1 apresenta a quantidade de erros cometidos em cada um desses artefatos em cada iteração. Percebe-se uma clara e abrupta redução de erros nos quatro artefatos. A primeira iteração ocorreu na fase de Iniciação. A segunda e terceira ocorreram na fase de Elaboração. As iterações restantes que constam no gráfico, ocorreram na fase de construção. É válido observar que o gráfico não apresenta todas as iterações do projeto porque a partir da sexta iteração todos os artefatos não mais apresentaram erros.

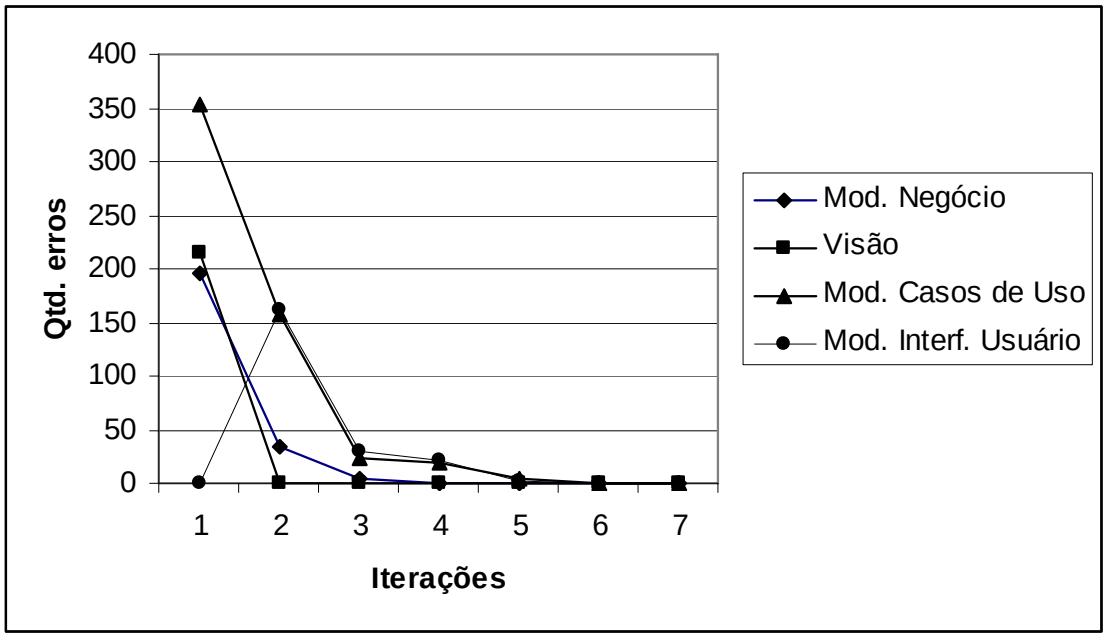


Gráfico 1 - Quantidade de erros identificados em artefatos criados pela ODS nas iterações

A forte queda da quantidade de erros já nas primeiras iterações foi bastante significativa, pois as fases de Iniciação e Elaboração serviram como escola para todas as equipes da ODS, preparando-se de tal forma que, na fase de Construção, os artefatos produzidos por ela já tinham um elevado padrão de qualidade antes da primeira verificação das iterações.

As listas de revisão dos artefatos foram ferramentas estratégicas na correção dos erros cometidos pela ODS. Sempre que uma revisão identificava erros em um artefato, este era devolvido juntamente com o relatório de revisão contendo as indicações e as evidências dos erros cometidos. Este forte controle da qualidade dos artefatos teve reflexo na FS1 que, em nenhuma iteração, precisou retornar a documentação recebida como insumos, para correção na ODS.

6.8.4.3 Análise dos Produtos da FS1

A FS1 executou, em todas as iterações, o escopo “D” (tabela 1 e figura 16) produzindo conseqüentemente os artefatos:

- Modelo de *Design;*
- Modelo de Dados;
- Modelo de Implantação.

A tabela 5 mostra, através dos resultados das verificações, que na fase de Iniciação, a FS1 produziu somente o modelo de implantação para oferecer uma visão geral da infra-estrutura necessária para tomada de decisões no planejamento do projeto. Contudo, nas demais fases, a FS1 produziu todos os três artefatos acima referidos.

Artefato	**Indicadores de qualidade**	**FASES / ITERAÇÕES / VERIFICAÇÕS**									
		Iniciação	**Elaboração**		**Construção**					**Transição**	
		I-1	**E-1**	**E-2**	**C-1**	**C-2**	**C-3**	**C-4**	**C-5**	**T-1**	**T-2**
Mod. *Design*	Total de erros	NA	2660	152	39	9	0	0	0	0	0
	Total de erros / caso de uso	NA	221,7	16,89	4,875	0,81	0	0	0	0	0
	Percentual de itens errados	NA	85,1	44,68	21,28	10,64	0	0	0	0	0
	Percentual de erros objetivos	NA	27,74	29,61	33,33	22,22	0	0	0	0	0
	Percentual de erros subjetivos	NA	72,26	70,39	66,67	77,78	0	0	0	0	0
Mod. Dados	Total de erros	NA	167	16	0	0	0	0	0	0	0
	Total de erros / caso de uso	NA	13,92	1,778	0	0	0	0	0	0	0
	Percentual de itens errados	NA	83,3	50	0	0	0	0	0	0	0
	Percentual de erros objetivos	NA	100	100	0	0	0	0	0	0	0
	Percentual de erros subjetivos	NA	0	0	0	0	0	0	0	0	0
Mod. Implantação	Total de erros	10	3	0	0	0	0	0	0	0	0
	Total de erros / nó-dispositivo	1,11	0,33	0	0	0	0	0	0	0	0
	Percentual de itens errados	50	21,43	0	0	0	0	0	0	0	0
	Percentual de erros objetivos	20	100	0	0	0	0	0	0	0	0
	Percentual de erros subjetivos	80	0	0	0	0	0	0	0	0	0
Totais de erros na iteração:		**10**	**2830**	**168**	**39**	**9**	**0**	**0**	**0**	**0**	**0**

Tabela 5 - Erros da FS1

Todos os indicadores de qualidade apresentados na tabela 5 também demonstram uma forte queda da freqüência de erros já nas três primeiras iterações, repetindo o que ocorreu na ODS. O modelo de *design* foi o artefato que mais apresentou erros, quando comparado com os demais artefatos.

Isso se deve ao baixo nível de granularidade dos itens de revisão do modelo de *design.* Analisando a lista de revisão do modelo de design no Anexo C, percebe-se que uma grande quantidade de itens refere-se às classes de *design*. A Tabela 6 mostra que, por ocasião da iteração E-1, a realização de 12 casos de uso resultou na identificação de 198 classes de *design* (fronteira, ativas, abstratas, interfaces e entidades), dando uma média de 16,5 classes por caso de uso. Esse foi o motivo do elevado índice de erros identificados: 2.260 erros, equivalente a 13,43 erros/classe de *design* e 221,67 erros/caso de uso realizado. Essa complexidade também foi refletida no tempo gasto para a estabilização do modelo[22]. O modelo de *design* necessitou de quatro iterações para atingir o nível zero de erros (Tabelas 5 e 6).

Indicadores de qualidade do Modelo de *Design*	**FASES / ITERAÇÕES / VERIFICAÇÕS**									
	Iniciação	**Elaboração**		**Construção**					**Transição**	
	I-1	**E1**	**E-2**	**C-1**	**C-2**	**C-3**	**C-4**	**C-5**	**T-1**	**T-2**
Total de erros:	NA	2660	152	39	9	0	0	0	0	0
Total de erros / caso de uso realizado:	NA	221,667	16,89	4,875	0,818	0	0	0	0	0
Total de erros / classe:	NA	13,4343	1,357	1,857	0,474	0	0	0	0	0
Percentual de itens com erros:	NA	85,1064	44,68	21,28	10,64	0	0	0	0	0
Percentual de erros objetivos	NA	27,7444	29,61	33,33	22,22	0	0	0	0	0
Percentual de erros subjetivos	NA	72,2556	70,39	66,67	77,78	0	0	0	0	0
Total de casos de uso realizados na iteração:	NA	12	9	8	11	13	13	9	5	0
Total de classes identificadas na iteração:	NA	198	112	21	19	22	20	14	8	0

Tabela 6 – Estatística de erros no modelo de *design*

O modelo de dados e o modelo de implantação tiveram baixos índices de erros e bastaram duas iterações para estabilizar o modelo. Isso se deu pelos seguintes motivos:

<u>Modelo de dados.</u>

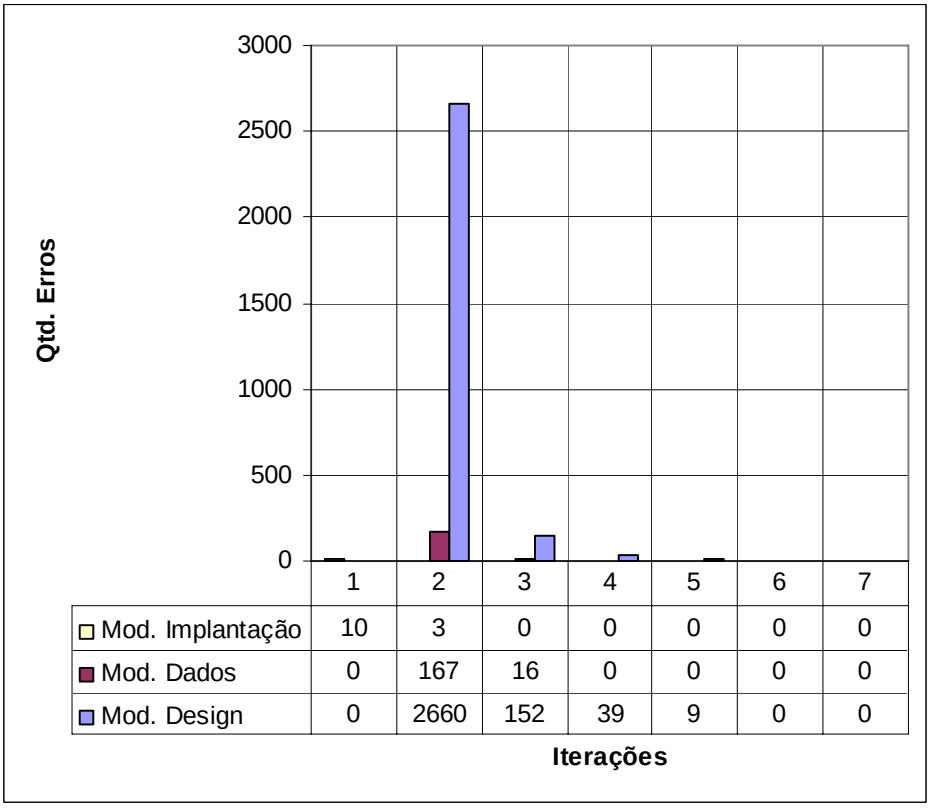
- Bastante popular entre todos os analistas.

[22] O termo "estabilização do modelo" no presente trabalho refere-se a um modelo que não apresenta erros por ocasião da aplicação da lista de revisão padrão daquele modelo.

- As entidades e atributos não necessitaram de descrições, porque as classes correspondentes, devidamente mapeadas com as entidades, já tinham essas descrições.

Modelo de implantação.

- Simplicidade, envolvendo poucos elementos na elaboração, o inverso do modelo de classes de *design.*



	1	2	3	4	5	6	7
Mod. Implantação	10	3	0	0	0	0	0
Mod. Dados	0	167	16	0	0	0	0
Mod. Design	0	2660	152	39	9	0	0

Gráfico 2 - Erros da FS1

O gráfico 2 oferece uma visualização melhor do que a tabela 5 das tendências de redução de erros ao longo do processo iterativo. Observa-se que, em cada iteração, a freqüência de erros cai até atingir zero erro. Observa-se também que após um modelo atingir sua estabilidade, esse nível de qualidade permanece até o fim do projeto. Isso se justifica pelo fato da fase de Elaboração proporcionar meios para que todos os modelos atinjam um elevado nível de estabilidade. Até o fim da fase de Elaboração, a equipe já passou por três iterações (uma na

Iniciação e duas na Elaboração), deixando-a com mais experiência. Isso mostra que a fase de Elaboração além de objetivar desenvolver uma arquitetura estável para o sistema, também cumpre o papel de amadurecer a equipe.

O que levou a fase de Elaboração alcançar seu objetivo de estabilizar os modelos foi o fato de priorizar os casos de uso de importância arquitetural para a definição da arquitetura.

Os critérios utilizados para tal foram:

- Realizar requisitos não-funcionais de elevado risco.
- Estender o modelo de domínio do sistema.

Conforme mostrado na tabela 7, dos 43 casos de uso identificados até o fim da fase de Elaboração (Tabela 7), foram priorizados 21 casos de uso de importância arquitetural, correspondendo a 48,84% do total de casos de uso já identificados até a ocasião. Esses 21 casos de uso foram totalmente realizados, proporcionando um modelo de classes constituído por 310 classes de *design*.

Totalizações	**FASES / ITERAÇÕES / VERIFICAÇÕS**									
	Iniciação	**Elaboração**		**Construção**					**Transição**	
	I-1	**E-1**	**E-2**	**C-1**	**C-2**	**C-3**	**C-4**	**C-5**	**T-1**	**T-2**
Total de casos de uso selecionados para a iteração em foco:	0	9	7	6	9	10	10	8	0	0
Total de casos de uso identificados na iteração em foco:	38	3	2	2	2	3	3	1	5	0
Total de casos de uso da tratados na iteração:	38	12	9	8	11	13	13	9	5	0
Total de casos de uso já arquitetados:		12	21							
Percentual de casos de uso de importância arquitetural:		48,84								
Total de casos de uso implementados:				8	19	32	45	54	59	59
Total geral de casos de uso (já identificados até o fim desta iteração):	38	41	43	45	47	50	53	54	59	59

Tabela 7 – Estatística de casos de uso

A partir da iteração C-3 nenhum erro foi identificado nos artefatos criados pela FS1, demonstrando que o modelo do negócio, documento de visão, modelo de

casos de uso, requisitos e modelo de interface do usuário foram suficientes para o desenvolvimento do modelo de *design*, modelo de dados e modelo de implantação estáveis.

A documentação usada como insumo (modelo do negócio, modelo de casos de uso, documento de visão, requisitos e modelo de interface do usuário) pela FS1 foi totalmente especificada em UML pela ODS. A partir dessa documentação a FS1 produziu o modelo de *design* e o modelo de dados sem nenhuma interferência da ODS ao longo do processo. Porém, para a produção do modelo de implantação a documentação usada como insumo não foi suficiente. Este fato ocorreu devido ao desconhecimento técnico da equipe do departamento de gestão de requisitos em assuntos relacionados à infra-estrutura e nas técnicas de distribuição e de modelagem de implantação.

Houve a tentativa de um maior detalhamento dos requisitos para dar suporte à FS1 na elaboração do modelo de implantação, porém percebeu-se que a própria elaboração do modelo daria menos trabalho do que elaborar textualmente requisitos suficientes para uma outra equipe elaborar o modelo de implantação. Além disso, a equipe de analistas de negócio e a de analistas de sistemas do departamento de gestão de requisitos não tinham o perfil adequado para a especificação de requisitos suficientes para que a FS1 pudesse elaborar, a partir deles, o modelo de implantação.

Com isso, ficou decidido que a elaboração do modelo de implantação seria feita pela ODS (não mais pela FS1), em paralelo ao levantamento dos requisitos na fase de iniciação e nas demais fases do processo. Assim, o escopo “NR” da ODS foi expandido para também produzir o modelo de implantação. Com isso, as atividades das disciplinas do escopo “NR” da ODS passaram a ser executadas paralelamente com duas atividades pertencentes à disciplina de Análise-*Design*, as quais tratam da elaboração do modelo de implantação [RUP2002]. Estas atividades foram:

- Análise Arquitetural / Desenvolver o Modelo de Implantação de Nível Superior.
- Refinar Arquitetura / Descrever Distribuição.

É válido observar que somente essas duas atividades foram realizadas em paralelo. As demais atividades da disciplina de Análise-*Design* só foram executadas após o recebimento da SS da ODS juntando todos os artefatos produzidos pelo escopo “NR” da ODS, agora também incluindo o modelo de implantação.

Desta forma, o escopo “NR” passa a produzir os seguintes artefatos:

- Modelo do Negócio.
- Modelos de Casos de Uso.
- Documento de Visão.
- Requisitos.
- Modelo de Interfaces do Usuário.

O escopo “D” passa a produzir os seguintes artefatos:

- Modelo de *Design.*
- Modelo de Dados.

Esta decisão trouxe a seguinte questão: quem irá executar essas duas atividades? Não poderá ser ninguém da FS1 porque existe a restrição dela estar situada em um local que inviabilize este contato com o arquiteto de software da empresa contratante. A solução foi a criação de uma nova OSI para o mesmo escopo “NR” endereçada ao departamento de análise-*design*.

Com isso o escopo “NR” passou a ser executado através de duas OSIs: uma para o departamento de gestão de requisitos como o objetivo de elaborar o modelo do negócio, modelo de casos de uso, documento de visão do sistema, requisitos, modelo de interface do usuário e o modelo de implantação. A

segunda OSI, endereçada ao departamento de análise-design com o objetivo de elaborar unicamente o modelo de implantação.

Além dos insumos acima descritos, a FS1 se utilizou do Guia de Análise-*Design*. Este guia foi criado na disciplina de Ambiente na fase de Iniciação, porém foi refinado ao longo do processo. A principal fonte de informação para o enriquecimento deste Guia foram os *feedbacks* obtidos nas verificações. A ocasião em que o Guia sofreu mais atualizações foi durante a fase de Elaboração. Essas atualizações eram feitas baseadas nos problemas identificados nas verificações dos produtos.

6.8.4.4 Análise dos Produtos da FS2

A FS2 ficou responsável pela execução do escopo “P” que corresponde à disciplina de Implementação. O modelo de *design* criado pela FS1 e o modelo de implementação criado pela FS2 tiveram uma abordagem compatível com o *Model Driven Architecture* (MDA) [OMG2003] sugerido pela OMG, o qual permite a separação da especificação das funcionalidades de um software, da especificação da implementação dessas funcionalidades. O modelo de *design* cuidou da especificação das funcionalidades do software em questão, enquanto que o modelo de implementação cuidou da especificação da implementação desse sistema. O primeiro não teve nenhuma abordagem relacionada à plataforma de implementação. O segundo é fortemente orientado à plataforma tecnológica utilizada.

O mapeamento entre os dois modelos foi feito pela FS2 que o fez a partir do modelo de *design* recebido como um dos insumos. O modelo de implementação foi desenvolvido para uma plataforma J2EE como tecnologia para implementação de software em camadas e utilização de componentes distribuídos e reutilizáveis, *TomCat* como *container* para classes JSP e *servlets* e *Hibernate* como *framework* para o mapeamento objeto-relacional.

Na fase de construção, já na primeira iteração, percebeu-se que a FS2 não estava sendo fiel à visão arquitetural do modelo de implementação desenvolvida na fase de Elaboração. Isso foi percebido nos testes de verificação (caixa-branca) aplicados nos componentes entregues à ODS.

Para solucionar o problema, optou-se pela subdivisão do processo de implementação em duas etapas porque a gerência do projeto quis garantir que o modelo de implementação tivesse total aderência ao modelo de *design,* garantindo que a FS2 iria implementar todos os aspectos arquitetônicos projetados. Em cada iteração, o modelo de implementação sofria seu incremento relativo ao conteúdo da iteração em foco, após o que o modelo era verificado pela ODS. As etapas foram:

- Criação do modelo de implementação, o qual precisava ser verificado pela ODS antes da codificação propriamente dita.
- Codificação dos componentes projetados no modelo de implementação.

A expectativa era que o modelo de implementação passasse a ter um *status* de importância para todos os desenvolvedores da FS2. O problema residia no perfil dos seus profissionais, os quais eram todos especializados em Java, mas com pouca experiência em modelagem visual, principalmente com uso de UML. Para homogeneizar os conhecimentos, foi ministrado um treinamento *on the job* a todos os profissionais da FS2, inclusive ao líder.

Assim, o processo foi modificado e ficou da seguinte forma:

1. FS2 recepciona a SS com os insumos produzidos pela FS1 e enviados pela ODS.
2. FS2 desenvolve o modelo de implementação.
3. ODS verifica o modelo de implementação.
4. FS2 codifica os componentes.
5. FS2 testa unitariamente os componentes.
6. FS2 entrega os componentes e modelo de implementação à ODS.

7. ODS executa testes de verificação funcionais.
8. ODS executa testes caixa-branca para verificar aderência ao modelo de implementação.

É válido notar que esse trecho do processo é totalmente suportado pelo processo operacional demonstrado na figura 19. Além da verificação do modelo de implementação, a ODS executou dois tipos de testes: teste funcional e teste caixa-branca. O primeiro usando os casos de teste para verificar se os componentes atendiam aos requisitos funcionais e não-funcionais. O segundo para verificar se o código construído era aderente ao modelo de implementação.

Devido à falta de software que automatizasse os testes caixa-branca, estas verificações foram feitas manualmente somente em uma amostra constituída por cerca de 20% das classes desenvolvidas na iteração em foco.

Comparando a quantidade de erros identificados nas iterações da fase de Elaboração entre o modelo de *design* (tabela 6) e o modelo de implementação (tabela 8), o segundo modelo apresentou uma quantidade de erros bem menor que o primeiro, embora o nível de granularidade dos dois modelos seja idêntico.

Lista de revisão	FASES / ITERAÇÕES									
	Iniciação	Elaboração		Construção					Transição	
	I-1	E1	E-2	C-1	C-2	C-3	C-4	C-5	T-1	T-2
Total de erros:	NA	48	9	2	1	0	0	0	0	0
Total de erros / caso de uso realizado:	NA	4	1	0,25	0,091	0	0	0	0	0
Total de erros / classe:	NA	0,186	0,06	0,074	0,04	0	0	0	0	0
Percentual de itens errados:	NA	78,947	63,16	35	20	0	0	0	0	0
Percentual de erros objetivos:	NA	81,25	88,89	50	0	0	0	0	0	0
Percentual de erros subjetivos:	NA	18,75	11,11	50	100	0	0	0	0	0
Total de casos de uso realizados na iteração:	NA	12	9	8	11	13	13	9	5	0
Total de classes de implementação identificadas na iteração:	NA	258	149	27	25	25	21	21	8	0

Tabela 8 - Estatística de erros no modelo de implementação

O motivo dessa diferença se encontra no fato de que o modelo de implementação é uma derivação do modelo de *design.* Todas as realizações dos casos de uso e dos requisitos não-funcionais já foram feitas e todas as decisões de arquitetura e de *design* já foram tomadas no modelo de *design.*

Devido ao padrão MDA [OMG2003] as decisões de arquitetura e de design são tomadas no modelo de design e as decisões de implementação são tomadas no modelo de implementação. Por isso, a elaboração do modelo de implementação consiste basicamente de copiar o modelo de design e fazer as devidas adaptações do design para adequar-se corretamente às tecnologias usadas para a implementação do sistema. A elaboração do modelo de implementação passa a ser uma atividade, de certa forma, simples e não demorada.

Com a validação do modelo de implementação imediatamente antes da codificação dos componentes, criou um fator psicológico bastante importante na equipe da FS2, tendo como conseqüência uma forte queda nos erros cometidos nos testes caixa-branca dos componentes, conforme demonstrado na tabela 9.

Testes executados	**FASES / ITERAÇÕES**									
	Iniciação	**Elaboração**		**Construção**					**Transição**	
	I-1	**E1**	**E-2**	**C-1**	**C-2**	**C-3**	**C-4**	**C-5**	**T-1**	**T-2**
Totais de erros em testes funcionais:	NA	NA	NA	48	49	35	21	12	28	37
Totais de erros em testes caixa-branca:	NA	NA	NA	22	2	0	0	0	0	0
Total de erros:	NA	NA	NA	70	51	35	21	12	28	37
Percentual de casos de testes com erros:	NA	NA	NA	76,19	53,84	35,35	24,13	17,91	18,18	14,62
Percentual de testes caixa-branca com erros	NA	NA	NA	78,57	7,14	0	0	0	0	0
Total de casos de uso tratados na iteração	NA	NA	NA	8	11	13	13	9	5	0
Total de casos de teste (funcionais):	NA	NA	NA	63	91	99	87	67	154	253
Total de itens de revisão (caixa-branca):	NA	NA	NA	28	28	28	28	28	28	28

Tabela 9 – Estatística de falhas identificadas nos teste funcionais e caixa-branca.

6.9 Comprovação das Hipóteses

A elaboração de todos os artefatos dentro da ODS para serem enviados para a FS1 e, na FS1 para serem enviados para a FS2, foi feita com todo o esmero com o objetivo de garantir a correta utilização da UML e das padronizações definidas nos guias de modelagem. Todos os elementos de todos os modelos construídos tiveram uma descrição conceitual buscando objetividade e não ambigüidade para oferecer um completo entendimento dos requisitos e especificações e assim eliminar a necessidade de comunicação indesejada e onerosa entre as três organizações: ODS, FS1 e FS2.

Esses objetivos foram alcançados, conforme demonstrado na seção anterior, por causa dos seguintes fatores:

- Gerente do projeto possuía um profundo conhecimento do RUP e da UML. Ele esteve constantemente atento a todos os aspectos relacionados à qualidade dos produtos intermediários (modelos, arquiteturas, visões, notação UML, Guia de Modelagem de Casos de Uso, Guia de Análise-Design e Guia de Implementação), bem como à completa e correta aplicação do processo de qualidade, desde a definição e refinamento das listas de revisão e casos de teste até a correta descrição das evidências de erros para agirem como *feedback* para as equipes de desenvolvimento dos artefatos.
- Processo de qualidade foi o grande responsável pela geração de todo o *feedback* necessário para a evolução das equipes e conseqüente melhoramento da qualidade final dos artefatos produzidos ao longo de todo o processo de desenvolvimento.
- Treinamento constante das equipes. Sempre que um treinamento se fazia necessário, a equipe carente passava por um treinamento "*on the job"* para uma correta aplicação da UML e do processo.

A seção 6.7 mostra como as hipóteses deverão ser comprovadas. Com base no exposto na seção 6.8, pode-se considerar que as quatro hipóteses são verdadeiras devido aos seguintes fatos:

- Hipótese - 1:
    - Os cenários com seus respectivos escopos foram planejados e distribuídos para a ODS, FS1 e FS2.
    - O planejamento usou um único tipo de cenário visando viabilizar os trabalhos nas FSs envolvidas.
- Hipótese - 2:
    - Os produtos gerados pela ODS para serem enviados para a FS1 foram:
        - Modelo de negócio constituído por um conjunto de casos de uso de negócio e suas respectivas realizações formadas por um modelo de domínio e um conjunto de diagramas de atividades descrevendo os processos de negócio relacionados aos casos de uso de negócio.
        - Documento de visão do sistema, apresentando a sentença dos problemas identificados, as necessidades dos envolvidos e um conjunto de características de software baseadas na [ISO9126] e em [LEF2000].
        - Modelo de casos de uso contendo:
            - Diagramas de casos de uso, descrições dos fluxos principais, alternativos e de exceção, pré e pós-condições.
            - Atores e suas respectivas descrições.
            - Empacotamento de casos de uso em subsistemas que foram bastante utilizados na definição dos subsistemas de *design* e de implementação.
        - Requisitos:

        - Requisitos não-funcionais baseados na [ISO9126] devidamente descritos de forma simples, clara e não ambígua.
        - Regras de negócio contendo uma descrição das restrições de negócio, condições, normas internas, legislação e políticas corporativas e departamentais que deveriam ser implementadas.
- Modelo de interface do usuário contendo:
    - Um conjunto de protótipos de formulários.
    - Conjunto de classes de análise (*boundaries*) especificando os formulários usados e respectivos atributos especificando os campos dos formulários com seus respectivos metadados.
- Modelo de implantação contendo:
    - Todos os nós e dispositivos da rede de computadores utilizados na implantação e distribuição do sistema.
    - Cada nó continha a descrição da sua configuração: processador, memória, *clock* do processador, capacidade de disco, sistema operacional, sistema gerenciador de banco de dados (para o servidor de banco de dados).
    - Em cada link continha a descrição do protocolo de comunicação utilizado e a especificação do cabo da rede.
- Guia de modelagem de casos de uso para ser usado nas três organizações (ODS, FS1 e FS2).
- Guia de análise-*design* para ser usado nas três organizações.
- Guia de programação para ser usado nas três organizações.
- Listas de revisão de produtos intermediários (veja Anexo C).

- Os produtos gerados pela FS1 são aderentes ao escopo definido para ela:
  - Modelo de *design* constituído por:
    - Realizações de casos de uso.
    - Subsistemas de *design* dispostos em camadas com suas respectivas interfaces, classes abstratas, classes ativas e classes de persistência.
    - Pacotes reunindo elementos de importância gerencial.
    - Diagramas de máquina de estados.
    - Diagramas de rastreabilidade mostrando o mapeamento entre os elementos de vários tipos de modelos.
  - Modelo de dados mapeados com as classes de persistência.
- Os produtos gerados pela FS2 são aderentes ao escopo definido para ela:
  - Modelo de implementação totalmente mapeado com modelo de *design*.
  - Componentes testados unitariamente.

- Hipótese - 3:
  - A FS1 usou como insumos os artefatos produzidos pela ODS e criou o modelo de *design* ao longo das iterações do processo. O modelo de *design* foi validado em cada iteração e usado como insumo pela FS2.
  - A FS2 usou o modelo de *design* como insumo para produzir em cada iteração, o modelo de implementação e os componentes testados unitariamente.
  - Todos os artefatos foram especificados usando a UML como notação, fato que proporcionou uma boa comunicação entre as três organizações sem a necessidade de reuniões complementares

para esclarecimentos de dúvidas e complementações dos requisitos.

- Os artefatos produzidos pelas duas FSs foram validados pelo departamento de qualidade da ODS, mostrando que os insumos para a produção desses artefatos tinham a qualidade necessária para tal.
- Em todas as iterações das fases de Construção e Transição os produtos de software construídos foram verificados internamente pela ODS, comprovando que as especificações criadas foram suficientes para tal.

- Hipótese - 4:
  - O processo utilizado no presente projeto apresentou os seguintes resultados:
    - O cenário definido para o presente projeto foi plenamente atendido pelo processo definido.
    - A mudança de cenário ao longo do projeto não foi testada porque isso iria alterar o escopo das duas FSs envolvidas, o que não foi aprovado pelo CTO, devido aos riscos envolvidos na mudança.
    - Embora o projeto não tenha testado mudanças de escopo e de FSs, o processo está plenamente preparado para dar suporte a esses tipos de eventos. Contudo o problema, não se encontra no processo mas na gestão do projeto e na viabilização legais e econômica dessas mudanças.
    - O processo efetuou todo o controle das interações entre as três organizações, bem como as atividades internas da ODS, tais como gestão do projeto, gestão da qualidade e gestão de configuração e mudança.

Com base no exposto ficam comprovadas as hipóteses propostas pelo presente projeto.

Capítulo 7 – Conclusões

O presente capítulo apresenta as conclusões que foram tiradas ao longo do projeto de pesquisa-ação. Pretende também apresentar problemas que ainda não foram resolvidos ficando como sugestão para futuras pesquisas. As conclusões são:

1. O ponto de divisão do trabalho realizado pela ODS e pela FS é variável, e essa variação pode ocorrer entre as fases e entre as iterações. Quem define o ponto de separação entre o trabalho da ODS e da FS é o escopo de um cenário.

2. O termo “escopo de uma fábrica de software” não é adequado, porque o escopo não está associado a uma FS e sim a uma SS. Uma SS tem um escopo que poderá ser executado por uma FS. Com isso, os termos “Fábrica de Projeto” e “Fábrica de Componentes” não são pertinentes, porque isso reflete um escopo. O termo que deve ser usado é simplesmente “Fábrica de Software”, que em um projeto poderá executar um escopo que envolve disciplinas de Modelagem de Negócio, Requisitos e Análise-Design e, em outro projeto, poderá estar envolvida somente com a disciplina de Implementação.

3. Do ponto de vista da FS, um escopo é uma unidade de trabalho que pode receber internamente, um tratamento de um pequeno projeto. Do ponto de vista da ODS, um escopo é uma porção bem definida do sistema em desenvolvimento, que deverá ser desenvolvida externamente por uma FS ou internamente por um departamento da ODS. Quando desenvolvida externamente por uma FS, a ODS sabe o que será produzido e quando será entregue. Assim, a ODS mantém um forte controle da qualidade e do

ritmo da produção, decidindo o que, quando e quem produz. Com isso a ODS sabe exatamente o que está ocorrendo dentro de cada FS envolvida. Esse controle da produção significa dizer que a ODS detém o controle do cronograma de execução do projeto, e o controle das entregas das FSs envolvidas.

4. Para que a FS1 possa atender às expectativas da ODS descrita no presente trabalho é necessário que:
   a. Ela possua uma unidade organizacional que recepcione as SSs originárias da ODS. Esta unidade deverá possuir uma equipe de analistas de sistemas com excelentes conhecimentos em UML e especialmente nas disciplinas de Modelagem do Negócio e de Requisitos para que possam validar todos os insumos recebidos para a produção dos artefatos solicitados.
   b. A FS1 possua uma equipe de produção com profundos conhecimentos na disciplina de Análise-*Design* para a criação do modelo de análise, modelo de projeto, modelo de implantação, modelo de dados e definição da arquitetura da aplicação porém sem o comprometimento de soluções de implementação seguindo as definições da MDA [OMG2003].
   c. Embora no projeto focado pela presente dissertação não tenha se utilizado de uma ferramenta de integração de modelos independentes de plataforma com a geração de modelos específicos para uma plataforma em específico, é altamente sugerido que a FS1 se utilize desses recursos tecnológicos com fins de agilizar o processo de desenvolvimento.
   d. É um fator crítico de sucesso que a FS1 possua uma célula de gerenciamento da qualidade para garantir que os modelos elaborados atendam aos requisitos entregues pela ODS como insumos, seguindo as orientações definidas no Guia de Análise-*Design.* Embora todos esses modelos venham a se submeter a um processo de validação pela ODS, a área de qualidade da FS1 certamente irá reduzir

substancialmente os erros dos modelos produzidos, elevando assim o desempenho do projeto como um todo.

5. Para que a FS2 possa atender às expectativas da ODS descrita no presente trabalho é necessário que:
   a. Igualmente à FS1, ela possua uma unidade organizacional que recepcione as SSs originárias da ODS para que faça o aceite dos modelos produzidos pela FS1 e validados pela ODS. Esta unidade deverá possuir uma equipe de analistas de sistemas com excelentes conhecimentos em UML e especialmente na disciplina de Análise-*Design* e Implementação.
   b. A FS2 possua uma equipe com conhecimento em UML para a leitura e entendimento dos modelos recebidos como insumos, para a produção do Modelo de Implementação que deverá ser validado pela ODS, antes da geração final do código.
   c. É fortemente sugerido que a FS2 disponha de tecnologia para a produção automática do Modelo de Implementação, que é um modelo dependente da plataforma, a partir dos modelos produzidos pela FS1, os quais foram concebidos independentemente da plataforma de implementação [OMG2003].
   d. A FS2 deverá ter uma unidade organizacional voltada para a garantida da qualidade, porém com dois enfoques. O primeiro para garantir a qualidade do Modelo de Implementação que deverá ser validado pela ODS, antes da codificação dos componentes. O segundo é possuir toda a infra-estrutura necessária para os testes dos componentes e dos *builds* construídos nas iterações.
   e. Deverá ter toda a tecnologia necessária para a produção automática de código a partir do Modelo de Implementação, restando ao programador somente a incumbência de complementar o código gerado. Isso irá garantir que o código produzido será totalmente aderente ao Modelo de Implementação validado pela ODS.

6. Controlar as interações significa dizer que a ODS é quem controla a execução do processo, definindo o início e o fim de cada escopo de uma FS. Quando o controle do processo fica sob o domínio da FS, a ODS assume uma postura passiva, porque ela não sabe exatamente o que a FS está trabalhando, ou seja, o trabalho que a FS está executando não foi disparado pela ODS.

7. Pode-se classificar o controle do processo de *software* como forte ou fraco. O controle do processo de software é fraco, quando a determinação de quando executar as atividades do processo fica a cargo da FS. O controle é forte quando essa determinação fica a cargo da ODS, a qual determina o ritmo de todas as equipes envolvidas no projeto, seja equipe interna ou externa (FS).

8. O modelo de processo de software executado na FS é indiferente para a ODS. Cada SS pode ser executada pela FS como se fosse um pequeno projeto, utilizando o modelo seqüencial clássico, o RUP ou qualquer uma metodologia ágil. Portanto, no contexto do presente trabalho, a FS pode ser considerada uma “caixa preta”.

9. Embora os mecanismos propostos permitam uma variação em cada iteração dos escopos da ODS e das FSs envolvidas no projeto, essa variação nem sempre é possível ser praticada, porque o envolvimento de uma FS em um projeto de desenvolvimento requer um planejamento interno da fábrica que pode significar contratação de mão de obra especializada para atender o projeto em questão, aquisição de servidores, computadores pessoais, software básico, sistema gerenciador de banco de dados, ferramentas de desenvolvimento, ajustes de processos e treinamento. Tudo isso envolve recursos financeiros que precisam ser computados na composição dos preços a serem cobrados à ODS. A

mudança de escopo ao longo do processo de desenvolvimento poderá afetar negativamente a FS subcontratada. Portanto, o planejamento dos cenários deverá ser feito logo no início da primeira iteração da fase de Iniciação. Este cenário deverá ser constante ao longo de todo o ciclo de vida do projeto. O único argumento de mudança de cenário é o não cumprimento de cláusulas contratuais relacionadas à qualidade e prazos de entrega por parte da FS. Se isso ocorrer, a alteração de cenários e conseqüentes escopos da ODS e/ou FS será plenamente suportada pelo processo definido pelo presente projeto.

10. O planejamento dos cenários de cada iteração requer a distribuição dos escopos para a ODS e para as FSs envolvidas no projeto. Essa distribuição dos escopos pode ser homogênea ou heterogênea.
    a. A distribuição de escopos é homogênea quando um determinado escopo (por exemplo, o escopo “D”) de um cenário é executado por uma única FS em todas as iterações do ciclo de vida do projeto. Do ponto de vista da FS, ela irá executar as mesmas atividades produzindo os mesmos artefatos em cada iteração do projeto. Isso permitirá um planejamento prévio da FS para se preparar para executar o escopo subcontratado.
    b. Uma distribuição de escopos é heterogênea quando um determinado escopo (por exemplo, o escopo “P”) de um cenário é executado por mais de uma FS em diferentes iterações do processo. Isso é plenamente possível ser executado na fase de Construção, com o objetivo de pulverizar os componentes a serem construídos em múltiplas FSs. Pode ser usada como uma estratégia de segurança em que os componentes não ficam concentrados em uma única FS. A condição para que esse tipo de planejamento possa ser factível é que seja feito logo após a identificação dos casos de uso na fase de Iniciação.

11. Com os mecanismos apresentados neste projeto de pesquisa, a ODS não mais precisa se limitar à utilização de FS próximas à ODS. O gerente do projeto pode buscar no mercado FSs especializadas em determinadas disciplinas e planejar cenários com escopos que envolvam mais de uma FS em função da necessidade e da fase na qual se encontra.

12. Na hipótese da ODS não ter recursos humanos para executar nenhuma disciplina operacional, o gerente pode planejar cenários com escopos executados por duas ou mais FS sem nenhuma participação operacional da ODS. Neste caso a ODS executaria somente as disciplinas gerenciais (Gestão de Configuração e Mudança, Gestão de Projeto e Gestão do Ambiente) e, ainda assim, manteria o controle do processo. O ato de manter a execução do processo operacional de controle das interações entre a ODS e as FSs envolvidas nos cenários não está relacionado ao fato de a ODS executar ou não algum tipo de escopo, porque o controle se encontra na atividade “Gerenciar a Iteração” pertencente à disciplina de Gestão do Projeto. Por isso, a ODS pode se abster de executar qualquer escopo e, ainda assim, manter o controle de todo o processo operacional.

13. As disciplinas gerenciais não constituem escopos, porque não são variáveis ao longo do processo. Elas sempre serão executadas em todas as iterações de todas as fases pela ODS e pelas FSs envolvidas, se elas utilizarem o RUP como modelo de processo.

14. Não foi possível medir o aumento da produtividade da equipe na execução de um mesmo escopo ao longo do processo. Porém a melhora da qualidade dos artefatos produzidos foi facilmente percebida através dos resultados obtidos nas verificações efetuadas pela equipe de gestão de qualidade. A fase de Elaboração foi considerada aquela onde a equipe amadureceu e se preparou para a fase de Construção. As equipes da ODS

e as duas FSs envolvidas conseguiram reduzir seus níveis de erros a próximo de zero até a segunda e última iteração da fase de Elaboração.

15. Por meio de uma análise dos volumes de erros identificados nas verificações dos artefatos criados pela ODS e pelas duas FSs envolvidas em cada iteração de cada fase, percebe-se que as primeiras iterações apresentaram um volume elevado de erros, porém esse volume sofreu uma brusca queda ao longo das duas iterações da fase de Elaboração. Os fatores que contribuíram para essa redução de erros foram:
    a. Treinamento ministrado em princípios de análise e projeto orientados a objetos, RUP e UML para as equipes da ODS e para as duas FSs. Esse treinamento buscou a homogeneização dos conceitos gerais sobre os assuntos focados.
    b. Treinamentos *on the job* dada às equipes na medida em que se fazia necessário. Os treinamentos *on the job* apresentaram maior eficácia do que o treinamento tradicional que foi ministrado para todas as equipes. Isso se deve ao fato de que o treinamento tradicional é amplo, abrangente e pouco específico. O treinamento *on the job* é direcionado a um grupo específico que apresenta uma deficiência também específica. Essa especificidade leva a um curso curto e planejado para resolver o problema daquele grupo.
    c. O processo de qualidade teve uma fundamental importância na redução dos erros identificados nos artefatos produzidos pela ODS e pelas duas FSs. Percebeu-se fortemente a certeza, por parte das equipes de produção na ODS e nas FSs, de que tudo que fosse produzido seria verificado pela equipe de qualidade, além da certeza da cobrança das correções desses erros ainda dentro da iteração em execução. A permanência dessa certeza ao longo de todo o processo proporcionou uma espécie de pressão na equipe para que os artefatos já fossem produzidos com qualidade já na primeira “rodada” de produção.

d. Constante atualização dos guias de modelagem que foram efetuadas com base nos relatórios de verificação dos artefatos. Os guias continham as listas de revisão que eram usadas pela equipe de qualidade da ODS.
e. Aumento da experiência de cada membro da equipe.
f. A elaboração da arquitetura da aplicação ocasionou a produção de “esqueletos” dos seguintes modelos: modelo de casos de uso, modelo de interface gráfica do usuário, modelo de *design,* modelo de dados, modelo de implantação e modelo de implementação. Esses “esqueletos” foram estabilizados durante as duas iterações da fase de Elaboração. Esses “esqueletos” foram usados como um guia ao longo da fase de Construção, a qual teve o papel de somente evoluir aqueles modelos através do seu detalhamento. Como a equipe já estava com boa experiência obtida na fase de Elaboração, adicionada à uma Arquitetura já totalmente estruturada, testada e estabilizada, a fase de Construção foi a que menos apresentou erros porque poucas estruturas haviam para se criar e quando isso ocorria seguia a Arquitetura já idealizada.

16. A lista de revisão foi a maior fonte de informações para a equipe do projeto, tanto da ODS com das duas FSs envolvidas.

17. Com base na influência dos treinamentos na redução dos erros identificados pela equipe de qualidade, conclui-se que, se todas as equipes fossem formadas por profissionais com experiência em OO, RUP e UML, a quantidade inicial desses erros seria fortemente reduzida. Isso causaria impacto na melhoria da qualidade dos produtos intermediários e finais, na redução de prazos e na conseqüente redução de custos.

18. Um dos maiores problemas encontrados e que ameaçou o sucesso do projeto foi a dificuldade de se encontrar no mercado, profissionais com os conhecimentos necessários em UML e RUP. É bastante difícil encontrar

profissionais disponíveis no mercado com pelo menos um projeto totalmente executado com uso do RUP e UML. Como a procura é grande, na medida em que uma ODS investe em treinamento para a formação de uma equipe preparada para trabalhar fluentemente com RUP e UML, o mercado busca os profissionais que se sobressaem na aplicação dessas técnicas, oferecendo-lhes melhores salários, benefícios e até posições de liderança em projetos desafiadores.

19. Todos os mecanismos de gerenciamento e controle propostos por este trabalho também podem ser aplicados em projetos que não usam FS externa. Uma ODS que possui uma estrutura organizacional matricial semelhante ao Escritório de Engenharia de Software proposto por esse trabalho também necessita dos mesmos formalismos de comunicação e controle.

20. O processo de configuração e mudança foi de importância estratégica na comunicação interna e externa à ODS, evitando a informalidade no envio de documentos de uma equipe para outra, além de deixar registrado todo o ciclo de vida de um artefato e sua trajetória por entre as unidades de produção na ODS e nas FSs envolvidas no processo produtivo.

21. A obtenção de uma boa produtividade da equipe e da criação de artefatos com qualidade não depende da presença física do gerente do projeto. Depende de mecanismos de controle e de medição que deixe claro para a equipe que tudo que é produzido é avaliado, medido e visto não somente pelo gerente do projeto, mas por toda a liderança do escritório de projetos e da ODS. No presente projeto os mecanismos utilizados foram: processo operacional, processo de qualidade e o processo de configuração e mudança. Infelizmente não foi possível a automação desses mecanismos, fato que iria facilitar o registro de todos os fatos ocorridos ao longo de todo o processo produtivo, inclusive dos tempos gastos para a execução de

cada tarefa realizada, produzindo relatórios atualizados em tempo real para toda a liderança da ODS.

22. A UML se revelou ser uma linguagem de modelagem com todos os atributos necessários para proporcionar uma perfeita comunicação entre equipes de projetos localizadas geograficamente separadas para efetuarem trabalhos colaborativos em projetos de *software*.

23. Em projetos que utilizam uma única equipe para desenvolver um sistema de *software* com uso do RUP como modelo de processo, as atividades das disciplinas em uma iteração são executadas simultaneamente [RUP2002]. Porém em projetos que utilizam uma ou mais FSs, essas disciplinas são executadas dentro de escopos, onde um escopo, executado por uma organização, produz artefatos que serão usados como insumos em outro escopo, o qual é executado por uma outra organização remotamente localizada. Por isso, esses escopos necessitam ser executados de forma seqüencial dentro de uma iteração.

24. Durante todas as iterações de todas as quatro fases do projeto não foi necessária nenhuma reunião entre a ODS e as duas FSs para resolver problemas de qualidade dos artefatos entregues às FSs como insumo, nem tampouco na entrega dos artefatos produzidos por essas fábricas após a conclusão das SSs. Todos os erros identificados pelo departamento de qualidade da ODS nos produtos entregues pelas duas fábricas foram completamente corrigidos (sem a necessidade de reuniões entre as duas organizações) através dos processos de qualidade e de configuração e mudança estabelecidos na ODS, ambos controlados pelo processo operacional definido no presente trabalho. É válido notar que o termo “eliminação de reuniões entre a ODS e FS” usado no presente trabalho refere-se somente no tocante a execução das disciplinas operacionais do

RUP, ou seja, Modelagem do Negócio, Requisitos, Análise-*Design,* Implementação, Teste e Implantação.

25. Os mecanismos propostos pelo presente trabalho nem sempre serão aceitos e usados pela liderança de todo e qualquer ODS. Sua aceitação requer que a ODS tenha vivenciado uma série de problemas ao longo de sua vida que a conduza às seguintes necessidades:
    a. Utilização de um modelo de processo incremental e iterativo.
    b. Subcontratação de FS para executar escopos formados por disciplinas operacionais do RUP.
    c. Eliminação de reuniões entre a ODS e a FS subcontratada para resolver problemas de qualidade da documentação entregue à FS como insumo para a produção dos artefatos solicitados.
    d. Controle de todo o planejamento e do processo produtivo em cada iteração ao longo de todo o ciclo de vida do projeto, tendo a FS como uma unidade “caixa preta” de produção.
    e. Controle da qualidade de todos os artefatos produzidos pela ODS e pelas FSs envolvidas no processo em cada iteração do projeto.

    Empresas que têm essas necessidades normalmente são aquelas que já viveram experiências negativas em *outsourcing* de todos os projetos de software e perceberam a importância do gerenciamento de todo o conhecimento envolvido nas soluções definidas na automação dos processos de negócio e no alinhamento aos objetivos estratégicos da empresa.

26. A conclusão final deste trabalho é que, o principal fator crítico de sucesso para que uma ODS, que se utiliza do RUP como modelo de processo, possa ter sucesso no uso desacoplado de uma ou mais FSs, são os recursos humanos tanto da ODS como das FSs envolvidas no projeto. Todos os mecanismos definidos neste trabalho são plenamente viáveis e adaptáveis para cada situação específica. Porém, de nada adianta se os

membros das equipes da ODS e das FSs envolvidas não tiverem bons conhecimentos em UML e boa redação. Esses dois elementos são inseparáveis e essenciais no estabelecimento da comunicação entre essas duas organizações. De nada adianta saber UML se o profissional não sabe descrever os conceitos e procedimentos necessários para a completa especificação dos itens da UML. Percebe-se claramente uma grande dificuldade por parte dos profissionais em se expressar de forma sucinta, precisa e não ambígua.

7.1 Sugestões de Pesquisas

O objetivo desta seção é apresentar uma lista de temas de pesquisas que foram identificadas ao longo da presente dissertação.

A. Caracterização qualitativa e quantitativa das ODS contratantes de FSs dos principais pólos de produção de *software* do Brasil quanto à estrutura organizacional, processos e demais mecanismos que elas usam no controle do processo produtivo.

B. Caracterização qualitativa e quantitativa do perfil dos profissionais de desenvolvimento de *software* no Brasil, no que se refere ao uso da UML e das nove disciplinas do RUP.

C. Definição de mecanismos de controle de interações entre uma ODS e desenvolvedores remotamente distribuídos.

D. O processo de configuração e mudança como um mecanismo de melhoramento da comunicação entre as equipes de uma estrutura matricial voltada para o desenvolvimento de *software*.

E. O escritório de engenharia de *software* como uma forma de implantação de uma estrutura matricial forte voltada para o desenvolvimento e manutenção de *software*.

F. Definição de técnicas que podem ser utilizadas para descrição de itens da UML de forma objetiva e isenta de ambigüidades.

G. Definição dos itens que devem compor o Acordo de Nível de Serviço entre a FS e a ODS, em um projeto que usa o RUP como modelo de processo e a UML como notação, com uma restrição de não haver comunicação entre essas duas organizações por ocasião da execução das disciplinas operacionais do RUP.

H. Por ocasião da execução do presente projeto não foi possível a medição do consumo de tempo na produção de artefatos UML na ODS e nas duas FSs envolvidas, bem como na produção dos artefatos de software (*builds*) em cada iteração ao longo de todo o ciclo de vida do projeto. Sugere-se um estudo de caso usando todos os mecanismos definidos no presente projeto, porém com o foco voltado para a medição do consumo de tempo gasto em cada atividade de cada disciplina de cada iteração de cada fase, tanto na ODS quanto na FS, relacionando o perfil profissional do executor das atividades como uma importante variável a ser observada.

I. Um dos mecanismos importantes sugeridos pelo presente projeto é o planejamento dos cenários com seus respectivos escopos da ODS e das FSs envolvidas no projeto. Contudo, a definição dos critérios que podem ser utilizados pelo Gerente do Projeto para a configuração desses escopos não foi tratada neste trabalho. Uma pesquisa objetivando a definição desses critérios seria de grande utilidade para a complementação dos mecanismos que a ODS pode lançar mão.

Referências

1. [AAE1997] Aaen, I; et al.; *The Software Factory: Contributions and Illusions*; Proceedings of the Twentieth Information Systems Research Seminar in Scandinavia, Oslo, 1997.

2. [AURÉLIO] *Novo Dicionário Eletrônico Aurélio* versão 5.0.

3. [BOE1988] Boehm, B. W.; *A Spiral Model of Software Development and Enhancement;* IEEE, 1988.

4. [BAR2004] Barcaui, André B.; Quelhas, Osvaldo; Perfil de Escritórios de Gerenciamento de Projetos em Organizações Atuantes no Brasil; Revista Pesquisa e Desenvolvimento Engenharia de Produção; N.2; p.38-53; julho/2004.

5. [BAT2006] Bartie, Alexandre; O que caracteriza uma verdadeira Fábrica de Teste? Artigo escrito para o *web site* http://imasters.uol.com.br em 30/08/2006: http://imasters.uol.com.br/artigo/4632/des_de_software/o_que_caracteriza_uma_verdadeira_fabrica_de_testes/

6. [BOO2005] Booch, G.; Rumbaugh, J.; Jacobson, I.; UML Guia do Usuário; Segunda Edição; Editora Campus; 2005.

7. [BRI2006] Brito, J; Metodologia para Gestão do Processo de Qualidade de Software para Incremento da Competitividade da Móbile S/A; Obtido na Internet: http://www.mct.gov.br/upd_blob/3318.pdf; em 19/11/2006.

8. [BRI2004] Brito, R.; et al.; Uma experiência na Implantação de Processos em Uma Fábrica de Software Livre; VI Simpósio Internacional de Melhoria de Processos de Software – SIMPRO; 2004.

9. [CAB2006] Cabral, D. M.; et al.; Criação Ágil de uma Fábrica de Software com Membros Distribuídos; Universidade Federal de Pernambuco (UFPE). Obtido na iternet em 15/11/2006;

10. [CES2003] César, R.; Fábrica de software: uma vocação nacional?; Publicado pela Computerworld-Terra (São Paulo) em 12/06/2003; Obtido na internent em 15/11/2006; http://www.siscorp.com.br/imprensa/computerworld02.htm?documento=24655&Area=51.

11. [CIR2003] Ciribelli, Marilda C.; Como Elaborar uma Dissertação de Mestrado Através de uma Pesquisa Científica; Rio de Janeiro; 7Letras; 2003.

12. [COS2006] Costa, B. G. S.; et al.; Fábrica de Software: Da definição às lições aprendidas; Centro de Informática da Universidade Federal de Pernambuco. Obtido na Internet em 15/11/2006.

13. [COS2003] Costa, I.; Contribuição para o aumento da qualidade e produtividade de uma fábrica de software através da padronização do processo de recebimento de serviços de construção de softwares; Tese apresentada à Escola Politécnica da Universidade de São Paulo para a obtenção de título de Doutor em Engenharia de Produção; São Paulo, 2003.

14. [FER2004] Fernandes, A. A.; Teixeira, D. S.; Fábrica de Software – Implantação e Gestão de Operações; São Paulo; Atlas, 2004.

15. [FIO1998] Fiorini, S. T.; Staa, A.; Baptista, R. M.; Engenharia de Software com CMM; Rio de Janeiro-RJ; Brasport;1998.

16. [GIL2006] Gil, A. C.; Como elaborar Projetos de Pesquisa; 4ª Edição; Editora Atlas; 2006.

17. [GRE2004] Greenfield, J. “O Caso das Fábricas de Software”; Microsoft Corporation, 2004.

18. [IEEE 610.12-1990] IEEE Standard Glossary of Software Engineering Terminology; STD 610.12-1990.

19. [IEEE 829-1998] IEEE Standard for Software Test Documentation; IEEE 829 1998;

20. [ISO9126] ISO/IEC FCD 9126-1.2: Information Technology - Software Product Quality - Part 1: Quality Model, 1998.

21. [JAC1999] Jacobson, I.; Booch, G.; Rumbaugh, J.; The Unified Software Development Process; Addison-Wesley; 1999.

22. [KRU2001] Kruchten, Philippe; The Rational Unified Process: An Introduction; Addison-Wesley; 2000; 5th Printing, June 2001.

23. [LEF2000] Leffingwell, Dean; Widrig, Don; Management Software Requirement; Addison Wesley, 2000.

24. [MAN2004] Mano, Cristiane; Os Caçadores de Projeto – A função deles é encontrar e implementar as melhores idéias dentro da empresa; Artigo publicado pela revista “Exame”; Editora Abril; Edição 17/08/2004.

25. [MAR2004] Martins, P. Ventura; Silva, A. Rodrigues; Comparação de Metamodelos de Processos de Desenvolvimento de Software; 5ª Conferência para a Qualidade nas Tecnologias de Informação e Comunicações; pg. 179-186, 2004.

26. [MEL2001] Mello F., Moacyr C. de; Gerenciamento de Subcontratações com o Rational Unified Process; Rational Software; 2001.

27. [MRT1990] Matin, J.; Information Engineering – Planning and Analysis; Book II; Prentice-Hall; 1990.

28. [MRQ2004-A] Marques, H. M; et al.; Adaptação de um Processo de Desenvolvimento para Fábrica de Software Distribuída; Centro de Informática – Universidade Federal de Pernambuco; 2004. Obtido na Internet em 15/11/2006. http://www.cin.ufpe.br/~in953/olds/fabricaUm_ideias04_RevisaoFinal.pdf;

29. [MRQ2004-B] Marques, H. M; et al.; Fábricas de Software e o Processo de Desenvolvimento Segundo a Experiência da FábricaUm; Centro de Informática – Universidade Federal de Pernambuco; 2004. Obtido na Internet em 15/11/2006; http://www.cin.ufpe.br/~in953/olds/relatorios/fabrica1.pdf;

30. [OMG2002] Meta-Object Facility (MOF) Specification, Versão 1.4; Abril 2002.

31. [OMG2003] MDA Guide Version 1.0.1; omg2003-06-01.

32. [OMG2005] Software Process Engineering Metamodel Specification; Versão 1.1; Janeiro; 2005; Obtido na Internet em 20/10/2006: http://www.omg.org/docs/formal/05-01-06.pdf.

33. [PEN2004] Pender, T.; UML a Bíblia; Editora Campus; 2004.

34. [PMI2004] Um Guia do Conjunto de Conhecimentos em Gerenciamento de Projetos – Guia PMBOK; 3ª edição; PMI 99-001-2004.

35. [PRE1995] Pressman, R.S.; Engenharia de Software; 3ª Edição; Makron Books, 1995

36. [PRE2002] Pressman, R. S.; Engenharia de Software; 5ª Edição; McGraw-Hill, 2002.

37. [REI2001] Reis, Christian; Caracterização de um Modelo de Processo para Projetos de Software Livre; Monografia apresentada ao Instituto de Ciências Matemáticas e de Computação para o Exame de Qualificação, como parte dos requisitos para a obtenção do título de Mestre na Área de Ciências da Computação e Matemática Computacional; São Carlos, SP; 2001.

38. [ROC2004] Rocha, T. A.; Adequação de Processos para Fábrica de Software; VI Simpósio Internacional de Melhoria de Processos de Software – SIMPROS; São Paulo-SP; 2004.

39. [RUP2002] Rational Unified Process®; Versão 2002.05.00;

40. [SIY2007] Siy, Harveye P.; et al.; Making the Software Factory Work: Lessons from a Decade of Experience; Obtido na Internet: http://mockus.us/papers/factory.pdf; em 10/02/2007.

41. [SOA2006] Soares, M. S.; Comparação entre Metodologias Ágeis e Tradicionais para o Desenvolvimento de Software; Unipac - Universidade Presidente Antônio Carlos, Faculdade de Tecnologia e Ciências de Conselheiro Lafaiete. Obtido da Internet em 15/11/2006.

42. [SOF2006] SOFTEX; MPS.BR – Melhoria do Processo de Software Brasileiro – Guia de Implementação Parte 4 – Nível D (Versão 1); 2006.

43. [YOU1990] Yourdon, E.; Análise Estruturada Moderna – Tradução da terceira edição americana; Rio de Janeiro; Editora Campus; 1990.

44. [ZAN2005] Zanchett, C. A, Barvinski; Metamodelagem MOF e sua aplicação para modelagem de sistemas imunológicos artificiais; Dissertação submetida à Universidade Federal de Santa Catarina como parte dos requisitos para obtenção do grau de Mestre em Ciência da Computação; Florianópolis, maio de 2005.

Anexo A

SS – Solicitação de Serviço				Número da SS:
Data emissão:	Data de envio para FS:	Data da validação da SS pela FS:	Data prevista entrega:	Data entrega:
Destinatário:				

Contatos	e-mail	Celular	Fone

Sistema:	Projeto:	Fase:	iteração:	Escopo:

Insumos	Observação

Artefato a ser produzido:	Data prevista início	Data início	Data prevista fim	Data fim

Data do evento	Evento ocorrido

Anexo B

OSI – Ordem de Serviço Interno	Número da OSI:

Data emissão:	Data de envio para destinatário:	Data da validação pelo destinatário:	Data prevista entrega:	Data entrega:

Departamento destinatário:

Equipe	Função	e-mail	Ramal

Sistema:	Projeto:	Fase:	iteração:	Escopo:

Insumos	Observação

Atividades a serem executadas:	Data prevista início	Data início	Data prevista fim	Data fim

Data do evento	Evento ocorrido	Atividade

Anexo C

Lista de revisão do Modelo do Negócio

Lista de revisão do Modelo do Negócio	O/S	Número de Erros	Observação
Atores de negócio	**NA**		
Os atores de negócio identificados são externos ao sistema?	S		
A descrição dos atores de negócio é clara, simples e não ambígua?	S		
Existe uma pasta exclusiva para atores de negócio?	O		
Existe um diagrama de atores de negócio?	O		
Existe uma pasta para os casos de uso de negócio?	O		
Casos de uso de negócio	**NA**		
Existe um diagrama de casos de uso de negócio?	O		
Os casos de uso de negócio representam um processo de negócio que produz um resultado de valor para o ator de negócio?	S		
Os casos de uso de negócio possuem uma descrição sucinta?	O		
As descrições sucintas são simples, claras e não ambíguas?	S		
Cada caso de uso possui seus fluxos de eventos modelados com um diagrama de atividades?	O		
O diagrama de atividades usa a notação UML corretamente?	S		
Cada atividade possui uma descrição simples, clara e não ambígua?	S		
O diagrama de atividades possui raias indicando atores, unidades organizacionais ou trabalhadores?	O		
O diagrama de atividades cobre o fluxo básico e todos as variantes do negócio e exceções?	S		
Cada caso de uso de negócio abriga seu próprio diagrama de atividades?	O		
Os casos de uso de negócio estão empacotados sob o critério de domínio de negócio?	O		
Existe um diagrama de pacotes de casos de uso mostrando as interdependências?	O		
Modelo de objetos de negócio	**NA**		
Existe um modelo de objetos de negócio?	O		
O modelo de objeto de negócio utilizou-se corretamente da notação UML?	O		
Os trabalhadores foram descritos de forma clara, simples e não ambígua?	O		
Cada trabalhador possui um diagrama mostrando o mapeamento com as atividades que ele realiza?	O		
Cada unidade organizacional possui um diagrama mostrando as atividades que ela realiza?	O		

Cada unidade organizacional possui uma descrição simples, clara e não ambígua?	S		
Modelo de domínio	**NA**		
Existe um modelo de domínio?	O		
Cada entidade de negócio esta descrita de forma simples, clara e não ambígua?	S		
O modelo de domínio está localizado em uma pasta específica.	O		
Cada entidade de negócio possui atributos?	O		
Os atributos estão descritos de forma clara, simples e não ambígua?	S		
O conjunto de atributos é suficiente para caracterizar adequadamente a entidade em foco?	S		
O modelo de domínio usa a notação UML corretamente?	O		
O modelo de domínio atende ao negócio?	S		
As entidades de negócio estão corretamente mapeadas com as regras de negócio?	S		
As entidades de negócio estão corretamente mapeadas com os requisitos?	S		
Total de erros:			

Lista de revisão do Documento de Visão do Sistema

Lista de revisão da Visão do Sistema	O/S	Num. Erros	Observação
O problema	**NA**		
Existe um diagrama de causas e efeitos (diagrama de problemas)?	O		
O diagrama de problemas mostra as corretas causas-raiz do problema?	S		
Cada causa-raiz do problema tem uma sentença do problema?	O		
Cada solução apresentada endereça sua respectiva raiz do problema?	S		
A solução está descrita numa linguagem voltada para o patrocinador?	S		
Os *stakeholdres* estão devidamente caracterizados?	O		
Existe uma pasta exclusiva para os *stakeholders*?	O		
Existe uma pasta exclusiva para os problemas?	O		
Existe uma lista de restrições?	O		
As restrições endereçam aspectos relacionados à tecnologia, finanças, prazos e RH?	S		
Necessidades	**NA**		
Existe uma pasta exclusiva para as necessidades?	O		
As necessidades estão mapeadas com o problema?	O		
As necessidades estão descritas de forma clara, simples e não ambígua?	S		
As descrições das necessidades usam uma linguagem de negócio?	S		
Existe um diagrama mapeando cada necessidade com seus respectivos stakeholders?	O		
Características	**NA**		
Existe uma pasta para as características do sistema?	O		
O conjunto de características descreve adequadamente o sistema?	S		
A abordagem das características funcional foi adequada?	S		
A abordagem das características de manutenibilidade foi adequada?	S		
A abordagem das características de confiabilidade foi adequada?	S		
A abordagem das características de usabilidade foi adequada?	S		
A abordagem das características de eficiência foi adequada?	S		
A abordagem das características de portabilidade foi adequada?	S		
Todas as características possuem atributos?	O		
O atributo relacionado a risco está com o valor adequado?	S		
O atributo relacionado à prioridade está com o valor adequado?	S		
O atributo relacionado a esforço está com o valor adequado?	S		

O atributo relacionado ao estágio de desenvolvimento está atualizado?	S		
O atributo relacionado ao impacto na arquitetura está com o valor adequado?	S		
O diagrama, mapeando as características com as necessidades, está correto?	S		
As regras de negócio estão descritas de forma clara, simples e não ambígua?	S		
As regras de negócio estão corretamente rastreadas com as características?	S		
Existe uma pasta para armazenar as regras de negócio?	O		
Documento de Visão	**NA**		
O documento de visão de visão é claro e de simples leitura?	S		
O documento de visão é obtido a partir do EA sem necessidade de edição?	O		
O documento de visão está disponibilizado no formato html?	O		
Total de erros:			

Lista de revisão do Modelo de Casos de Uso

Lista de revisão do Modelo de Casos de Uso	O/S	Num. Erros	Observação
Existe no EA uma pasta de atores reunindo todos os atores?	O		
Todos os atores estão descritos de forma simples, clara e não ambígua?	S		
Os nomes dos atores são aderentes ao padrão? (*)	S		
Existe um diagrama de atores reunindo todos os atores?	O		
Existe um diagrama com a visão geral de casos de uso?	O		
Todos os casos de uso existentes estão no diagrama da visão geral do caso de uso?	O		
Todos os casos de uso estão empacotados?	O		
Existe um diagrama de casos de uso para cada pacote?	O		
Existe um diagrama de interdependência de pacotes?	O		
Os pacotes se encontram com baixo acoplamento?	S		
Cada caso de uso possui um diagrama mostrando sua rastreabilidade?	S		
O caso de uso produz um resultado de valor do ponto de vista do ator?	S		
Os nomes dos casos de uso são aderentes ao padrão?	O		
Atores que iniciam possuem associação com seta?	O		
Atores envolvidos não possuem associação com seta?	O		
O caso de uso possui sua descrição sucinta?	O		
A descrição sucinta é clara, simples e não ambígua?	S		
O atributo relacionado a risco está com o valor adequado?	S		
O atributo relacionado a prioridade está com o valor adequado?	S		
O atributo relacionado a esforço está com o valor adequado?	S		
O atributo relacionado ao estágio de desenvolvimento está atualizado?	S		
O atributo relacionado ao impacto na arquitetura está com o valor adequado?	S		
O diagrama que mapeia os casos de uso às características está correto?	S		
Precondição é clara, simples e não ambígua?	S		
Pós-condição é simples, clara e não ambígua?	S		
Os casos de uso possuem fluxo básico?	O		
As descrições dos fluxos básicos são simples, clara e não ambíguas?	S		
O início e o fim do fluxo básico estão bem definidos?	S		
A inclusão de casos de uso nos fluxos atende aos padrões?	O		
Os pontos de extensão atendem aos padrões?	O		
A indicação dos fluxos alternativos atende aos padrões?	O		
A Descrição dos fluxos alternativos são claros, simples e não ambíguos.	S		
Existe um fluxo alternativo para cada variação do negócio?	S		
A descrição dos fluxos indica os dados que são trocados entre o ator e o sistema?	O		
Os casos de uso foram devidamente fatorados?	S		

Os relacionamentos de inclusão estão corretos?	S		
Os relacionamentos de *extend* estão corretos?	S		
As rastreabilidades entre casos de uso e requisitos não-funcionais são adequadas e completas?	S		
As rastreabilidades entre casos de uso e regras de negócio são adequadas e completas?	S		
Os requisitos estão descritos de forma clara, simples e não ambígua?	S		
Os requisitos estão localizados em uma pasta específica?	O		
As regras de negócio estão descritas de forma clara, simples e não ambígua?	S		
As regras de negócio estão localizadas em uma pasta específica?	O		
Todas as regras de negócio estão associadas a, pelo menos, um caso de uso?	O		
Na descrição dos casos de uso, os termos de negócio de importância para o sistema estão em itálico?	S		
Todos os termos em itálico encontrados na descrição e nos fluxos de eventos estão descritos no glossário?	O		
Todas as entidades de negócio estão em itálico e definidas no glossário?	S		
As descrições dos itens do glossário são claras, simples e não ambíguas?	S		
Total de Erros:			

Lista de revisão do Modelo de Interface do Usuário

Lista de revisão do Modelo de Interface Usuário	O/S	Num. Erros	Observação
Cada caso de uso tem uma encenação no modelo de análise?	O		
Cada encenação tem um diagrama mostrando a rastreabilidade desde o caso de uso até as classes de fronteira?	O		
Cada encenação possui um fluxo de eventos?	O		
O fluxo de eventos possui todas as indicações necessárias e suficientes para a criação do protótipo?	S		
As encenações são consistentes com a descrição dos casos de uso?	S		
Cada encenação possui um diagrama de classe de fronteira mostrando a hierarquia de agregações?	O		
O conjunto de classes de fronteira oferece uma boa solução de navegabilidade?	S		
Os requisitos de usabilidade estão rastreados para as encenações e classes de fronteira?	O		
O conjunto de classes de fronteira realiza adequadamente os requisitos de usabilidade?	S		
Cada tela (janela) está modelada com uma classe de fronteira?	O		
Cada entidade de negócio está modelada com uma classe de fronteira?	S		
As classes de fronteira estão descritas de forma clara, simples e não ambígua?	S		
As classes de fronteira possuem operações para implementar suas responsabilidades?	S		
Os campos das telas estão especificados como atributos nas classes de fronteira?	O		
Cada atributo das classes de fronteira está com todos seus metadados descritos?	O		
Os metadados dos atributos especificam as restrições, domínios e regras de negócio?	S		
As classes de fronteira possuem operações para implementar suas responsabilidades?	O		
As descrições das operações das classes de fronteira são simples, claras e não ambíguas?	S		
Cada tela possui um protótipo?	O		
Existe um mapeamento (realização) entre as classes de fronteira e os respectivos protótipos.	O		
Existe um mapeamento (realização) entre cada encenação e os protótipos envolvidos?	O		
Total de erros:			

Lista de revisão do Modelo de ***Design***

Lista de revisão do Modelo de *Design*	O/S	Num. Erros	Observação
Subsistemas e pacotes.	**NA**		
O “particionamento” da camada de apresentação em subsistemas segue o padrão?	S		
O “particionamento” da camada de negócio em subsistemas segue o padrão?	S		
Cada subsistema possui pelo menos uma interface?	O		
Cada interface possui pelo menos uma classe que a implementa?	O		
Cada pasta (camada, subsistema, pacote) possui um diagrama contendo seu conteúdo?	O		
Os subsistemas estão arquitetados de forma a ter baixo acoplamento?	S		
Cada subsistema e suas interfaces estão dentro de um pacote com o mesmo nome do subsistema?	O		
Cada subsistema e suas respectivas interfaces estão relacionados por uma "realização"?	O		
Todas as classes dentro dos subsistemas têm visibilidade privada?	O		
Classes	**NA**		
As classes de fronteira e de controle estão localizadas na camada de apresentação?	S		
Todas as classes de entidade se encontram localizadas na camada de negócio?	S		
Todas as Classes dentro dos pacotes e subsistemas têm visibilidade que promove o encapsulamento?	O		
Todas as classes possuem uma descrição sumária?	O		
Existe algum tipo de redundância de classes ou conceitos?	S		
O conceito de cada classe segue o padrão?	S		
O conceito de cada classe está claro, simples e não ambíguo?	S		
O conceito de cada atributo é claro, simples e não ambíguo?	S		
Existe redundância entre os atributos?	S		
Os atributos têm visibilidade "privada"?	O		
Os atributos estão com seus *types* adequados?	S		
Os atributos estão com suas multiplicidades corretas?	S		
As descrições das operações são claras, simples e não ambíguas?	S		
A descrição do comportamento da operação é exata?	S		
As operações têm visibilidades que promovem o encapsulamento?	O		
As assinaturas das operações estão completas?	O		
As assinaturas das operações estão descritas nos padrões da UML?	O		
Associação	**NA**		

As associações estão refletidas corretamente com *types* de atributos?	O		
As associações estão com as multiplicidades corretas?	S		
Cada associação tem suas multiplicidades dos dois lados?	O		
As multiplicidades estão corretas?	S		
As navegabilidades estão coerentes com os atributos de referência?	O		
Os papeis estão representados adequadamente?	S		
Os nomes das associações são adequados?	S		
As associações entre o pacote cliente e a interface servidora é através de uma dependência?	O		
A dependência é do pacote em relação à interface?	O		
Realização de Casos de Uso?	**NA**		
Cada caso de uso possui sua realização?	O		
Cada realização possui pelo menos um diagrama de interação realizando o caso de uso?	O		
O conjunto de diagramas de interação cobre todos os cenários significativos de cada caso de uso?	S		
Cada mensagem nos diagramas de interação está especificada segundo o padrão da UML?	O		
Cada mensagem em diagramas de interação corresponde a uma operação na classe servidora?	O		
Cada realização possui um diagrama de atividades suplementando e agrupando todos os cenários do caso de uso?	O		
O diagrama de atividades possui uma raia para cada ator que interage com o caso de uso?	O		
As atividades executadas no diagrama de atividades se encontram em um nível lógico atômico?	S		
A notação UML no diagrama de atividades está correta?	O		
Cada realização possui um diagrama de classes com uma visão da sua realização?	O		
Este diagrama de classe apresenta todas as classes envolvidas em todos os cenários?	O		
A notação UML no diagrama de classes está correta?	O		
Total de erros:			

Lista de revisão do Modelo de Dados

Lista de revisão do Modelo de Dados	O/S	Num. Erros	Observação
Cada classe de entidade está mapeada para pelo menos uma entidade do modelo de dados?	O		
Todos os atributos das classes de entidade têm um correspondente no modelo de dados?	O		
Os *types* dos atributos do modelo de dados são coerentes com os do modelo de classes?	O		
As associações N-N no modelo de classes estão mapeadas para uma entidade associativa no modelo de dados?	O		
Os relacionamentos no modelo de dados têm cardinalidades compatíveis com as multiplicidades no modelo de classes?	O		
As integridades referenciais estão corretas?	O		
Total de erros:			

Lista de revisão do Modelo de Implantação

Lista de revisão do Modelo de Implantação	O/S	Num. Erros	Observação
Cada hardware, com capacidade de processamento e de memória, está modelado com um nó?	O		
Cada nó possui uma descrição da configuração (hardware e software) e sua localização geográfica?	O		
O nome do nó expressa seu papel na topologia do sistema?	S		
Existe um diagrama mapeando os subsistemas que deverão ser instalados em cada nó do sistema?	O		
Cada hardware sem capacidade de processamento está modelado como um dispositivo (nó estereotipado)?	S		
O nome do dispositivo expressa corretamente sua função no sistema?	S		
Cada dispositivo possui uma descrição simples, clara e não ambígua da sua função no sistema?	S		
A descrição da sua capacidade ou largura da banda de cada conector está correta?	S		
Cada conector informa o protocolo correto de comunicação a ser utilizado?	S		
O modelo de implantação apresentado cobre todos os nós e dispositivos que o sistema necessita?	S		
Diferentes localidades geográficas estão representadas por uma fronteira reunindo no seu interior todos os nós e dispositivos?	O		
Os requisitos de taxa de transferência entre os nós foram atendidos?	O		
As possibilidades de gargalo entre os nós foram consideradas?	O		
O volume de espaço livre na utilização da CPU foi identificado?	O		
Total de erros:			

Lista de revisão do Modelo de Implementação

Lista de revisão do Modelo de Implementação	O/S	Num. Erros	Observação
Modelo de Implementação			
Todos os subsistemas dos modelos de projeto e os componentes do modelo de implementação estão mapeados entre si?	O		
Cada camada definida no modelo de *design* tem uma correspondente no modelo de implementação?	O		
Toda e qualquer interação entre elementos de camadas diferentes é feita através de interfaces?	O		
Toda e qualquer interação entre diferentes componentes sempre é feita por meio de interfaces?	O		
A composição dos componentes permite uma distribuição equilibrada de trabalho entre os programadores?	S		
Algum componente de uma camada depende de outro pertencente a alguma camada superior?	O		
Cada camada possui um diagrama de componentes?	O		
Cada componente possui um diagrama de classes mostrando as classes que encapsula?	O		
Cada componente possui um conjunto de diagramas de interação mostrando seu comportamento?	O		
Cada build forma um conjunto coeso de subsistema passível de ser integrado?	S		

Anexo D

Estatística de erros identificados em cada revisão dos produtos intermediários

Artefato	Indicadores de erros	FASES / ITERAÇÕES / VERIFICAÇÕS																	
		Iniciação			Elaboração					Construção								Transição	
		I-1			E-1			E-2		C-1		C-2		C-3		C-4	C-5	T-1	T-2
		1	2	3	1	2	3	1	2	1	2	1	2	1	2	1	1	1	1
Mod. Negócio	Total de erros	196	46	0	34	0	0	4	0	0	0	0	0	0	0	0	0	0	0
	Total erros / caso de uso neg.	12,3	2,88	0	6,8	0	0	2	0	0	0	0	0	0	0	0	0	0	0
	Percentual de itens errados	57,57	33,33	0	33,33	0	0	9,09	0	0	0	0	0	0	0	0	0	0	0
	Percentual de erros objetivos	15,3	8,69	0	8,82	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Percentual de erros subjetivos	84,7	91,31	0	91,18	0	0	100	0	0	0	0	0	0	0	0	0	0	0
Visão	Total de erros	215	32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Total de erros / característica	4,67	0,69	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Percentual de itens errados	77,78	30,56	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Percentual de erros objetivos	11,63	3,12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Percentual de erros subjetivos	88,37	96,88	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Mod. Caso Uso	Total de erros	354	48	0	157	20	0	24	0	20	0	5	0	1	0	0	0	0	0
	Total de erros / caso de uso	9,31	1,26	0	13,08	1,67	0	2,67	0	2,5	0	0,46	0	0,08	0	0	0	0	0
	Percentual de itens errados	62,2	30	0	40	15	0	17	0	25	0	8,3	0	2,1	0	0	0	0	0
	Percentual de erros objetivos	5,93	0	0	18	10	0	4,2	0	5	0	0	0	0	0	0	0	0	0
	Percentual de erros subjetivos	94,07	100	0	82	90	0	95,8	0	95	0	100	0	100	0	0	0	0	0
Mod. Interf. Usuário	Total de erros	NA	NA	NA	161	26	0	29	0	22	0	2	0	0	0	0	0	0	0
	Total de erros / caso de uso	NA	NA	NA	13,42	2,17	0	3,22	0	2,75	0	0,18	0	0	0	0	0	0	0
	Percentual de itens errados	NA	NA	NA	61,9	38,1	0	42,86	0	38,1	0	9,52	0	0	0	0	0	0	0
	Percentual de erros objetivos	NA	NA	NA	30,43	0	0	3,45	0	22,73	0	0	0	0	0	0	0	0	0
	Percentual de erros subjetivos	NA	NA	NA	69,57	100	0	96,55	0	77,27	0	100	0	0	0	0	0	0	0
Mod. *Design*	Total de erros	NA	NA	NA	2660	454	0	152	0	39	0	9	0	0	0	0	0	0	0
	Total de erros / caso de uso	NA	NA	NA	221,7	37,83	0	16,89	0	4,88	0	0,82	0	0	0	0	0	0	0
	Percentual de itens errados	NA	NA	NA	85,1	38,3	0	44,68	0	21,28	0	10,64	0	0	0	0	0	0	0

	Percentual de erros objetivos	NA	NA	NA	27,74	15,42	0	29,61	0	33,33	0	22,22	0	0	0	0	0	0	0
	Percentual de erros subjetivos	NA	NA	NA	72,26	84,58	0	70,39	0	66,67	0	77,78	0	0	0	0	0	0	0
Mod. Dados	Total de erros	NA	NA	NA	167	26	0	16	0	0	0	0	0	0	0	0	0	0	0
	Total de erros / caso de uso	NA	NA	NA	13,92	2,167	0	1,778	0	0	0	0	0	0	0	0	0	0	0
	Percentual de itens errados	NA	NA	NA	83,3	33,3	0	50	0	0	0	0	0	0	0	0	0	0	0
	Percentual de erros objetivos	NA	NA	NA	100	100	0	100	0	0	0	0	0	0	0	0	0	0	0
	Percentual de erros subjetivos	NA	NA	NA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Mod. Implantação	Total de erros	10	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Total de erros / nó-dispositivo	1,11	0	0	0,33	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Percentual de itens errados	50	0	0	21,43	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Percentual de erros objetivos	20	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Percentual de erros subjetivos	80	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Mod. Implementação	Total de erros	NA	NA	NA	48	8	0	9	0	2	0	1	0	0	0	0	0	0	0
	Total de erros / caso de uso	NA	NA	NA	4	0,667	0	1	0	0,25	0	0,091	0	0	0	0	0	0	0
	Percentual de itens errados	NA	NA	NA	78,95	0	0	63,2	0	35	0	20	0	0	0	0	0	0	0
	Percentual de erros objetivos	NA	NA	NA	81,25	0	0	88,9	0	50	0	0	0	0	0	0	0	0	0
	Percentual de erros subjetivos	NA	NA	NA	18,75	0	100	11,1	0	50	0	100	0	0	0	0	0	0	0
Totais de erros por revisão		**775**	**126**	**0**	**3230**	**534**	**0**	**234**	**0**	**83**	**0**	**17**	**0**	**1**	**0**	**0**	**0**	**0**	**0**
Totais de erros por iteração		**775**			**3230**			**234**		**83**		**17**		**1**		**0**	**0**	**0**	**0**
Totais de erros por fase		**775**			**3464**					**101**								**0**	**0**