



UFPA

ANÁLISE E PROJETO ORIENTADO A OBJETOS USANDO UML E O PROCESSO UNIFICADO

RODOLFO MOACIR SEABRA JÚNIOR

1º SEMESTRE / 2001

**CENTRO DE CIÊNCIAS EXATAS E NATURAIS
UNIVERSIDADE FEDERAL DO PARÁ
CAMPUS UNIVERSITÁRIO DO GUAMÁ
BELÉM-PARÁ**

UNIVERSIDADE FEDERAL DO PARÁ
CENTRO DE CIÊNCIAS EXATAS E NATURAIS
CURSO DE CIÊNCIA DA COMPUTAÇÃO

RODOLFO MOACIR SEABRA JÚNIOR

**ANÁLISE E PROJETO ORIENTADO A OBJETOS
USANDO UML E O PROCESSO UNIFICADO**

TRABALHO SUBMETIDO AO COLEGIADO DO
CURSO DE CIÊNCIA DA COMPUTAÇÃO PARA
OBTENÇÃO DO GRAU BACHAREL EM
CIÊNCIA DA COMPUTAÇÃO.

Belém
2001

“ANÁLISE E PROJETO ORIENTADO A OBJETOS USANDO UML E O PROCESSO UNIFICADO”

Este Trabalho foi julgado em 10/08/2001 adequado para obtenção do Grau de Bacharel em Ciência da Computação, e aprovado na sua forma final pela banca examinadora que atribuiu o conceito _____.

M.Sc. Alfredo Braga Furtado
**ORIENTADOR E COORDENADOR DO CURSO DE CIÊNCIA
DA COMPUTAÇÃO**

MEMBRO DA BANCA EXAMINADORA

MEMBRO DA BANCA EXAMINADORA

MEMBRO DA BANCA EXAMINADORA

A minha mãe (in memoriam),
por todo o amor e carinho que me deu.

AGRADECIMENTOS

À Deus, divino criador, fonte de sabedoria, pela vida e por ter me proporcionado mais essa conquista.

À todos os amigos do peito que desde longa data sempre se fizeram presentes não só nas horas de festa, mas também nos momentos mais difíceis dessa caminhada, em especial ao Marcelo.

À minha namorada Cimélia, pelos ótimos momentos de convivência e pelas demonstrações de amizade.

À todos os mestres e professores da Universidade Federal do Pará que participaram dessa caminhada, em especial ao meu orientador M.Sc Alfredo Furtado, pela competência, incentivo e amizade no decorrer não só da elaboração deste trabalho, como em toda essa jornada acadêmica.

Aos meus amigos da ELETRONORTE, em especial a Eng^a. Luciana e aos companheiros de trabalho Wellington, Leonardo, Juarez e Norberto.

Aos meus inicialmente colegas de Universidade, e depois amigos Aroldo Cristiano, José Alex, Augusto Judásio, Herivelton e, em especial, Roberto Moura.

E à todos aqueles que direta ou indiretamente prestaram sua parcela de contribuição na elaboração deste trabalho.

“No fim tudo dá certo, se ainda não deu é porque ainda não é o fim.”

Fernando Pessoa

LISTA DE ABREVIações

LACEN	Laboratório Central
GL	Gerente do Lacen
GQL	Gerente da Qualidade do Lacen
GTL	Gerente Técnico do Lacen
PRIE	Procedimento de recepção, identificação e manuseio de Itens de Ensaio
RT	Relatório Técnico
SCOD	Sistema de Custos, Orçamentos e Despesas
SCSQ	Sistema de Controle do Sistema da Qualidade
SGSI	Sistema de Gestão de Serviços e Instrumentos
SIGP	Sistema de Gestão de Pessoal
SIPE	Sistema de Planejamento Estratégico
SS	Solicitação de Serviço

LISTA DE FIGURAS

Figura 2.1	- Classe Funcionario.....	5
Figura 2.2	- Instância da Classe Funcionário.....	6
Figura 2.3	- Caso de Uso.....	6
Figura 2.4	- Mensagens.....	7
Figura 2.5	- Estado.....	8
Figura 2.6	- Estado e seus Subestados.....	8
Figura 2.7	- Dependência.....	9
Figura 2.8	- Dependências entre Classes.....	9
Figura 2.9	- Generalização.....	10
Figura 2.10	- Generalização entre Classes.....	10
Figura 2.11	- Associação.....	11
Figura 2.12	- Agregação.....	12
Figura 2.13	- Atributo de Ligação ou Classe de Associação.....	12
Figura 2.14	- Diagrama de Classes.....	14
Figura 2.15	- Diagrama de Objetos.....	15
Figura 2.16	- Diagrama de Casos de Uso.....	17
Figura 2.17	- Diagrama de Seqüências.....	19
Figura 2.18	- Diagrama de Atividades.....	22
Figura 2.19	- Diagrama de Gráfico de Estados.....	24
Figura 3.1	- A vida de um Processo em Ciclos.....	30
Figura 3.2	- Ciclo de Vida do Processo Unificado.....	31
Figura 3.3	- Um ciclo com suas fases e iterações.....	32
Figura 3.4	- Fluxo de Requisitos.....	33
Figura 3.5	- Fluxo de Análise.....	35
Figura 3.6	- Fluxo de Projeto.....	37
Figura 3.7	- Fluxo de Implementação.....	39
Figura 3.8	- Fluxo de Teste.....	40
Figura 4.1	- Tela Principal do SIGLacen.....	55
Figura 4.2	- Diagrama Principal de Casos de Uso.....	59
Figura 4.3	- Diagrama de Casos de Uso 'Refinamento do Caso de Uso <u>Realiza Procedimento de Contratação</u> '.....	60

Figura 4.4	- Diagrama de Casos de Uso ‘Refinamento do Caso de Uso <u>Abre SS</u> ’.....	61
Figura 4.5	- Diagrama de Casos de Uso ‘Refinamento do Caso de Uso <u>Conclui SS</u> ’.....	62
Figura 4.6	- Diagrama de Classes do Modelo de Análise. Fase de Concepção.....	63
Figura 4.7	- Diagrama de Classes do Modelo de Projeto. Fase de Concepção.....	64
Figura 4.8	- Diagrama de Casos de Uso ‘Refinamento do Caso de Uso <u>Coleta informações do Cliente</u> ’.....	65
Figura 4.9	- Diagrama de Casos de Uso ‘Refinamento do Caso de Uso <u>Emite o documento Contrato para a Execução do Serviço</u> ’.....	66
Figura 4.10	- Diagrama de Casos de Uso ‘Refinamento do Caso de Uso <u>Abre SS para o Item Recebido</u> ’.....	67
Figura 4.11	Diagrama de Casos de Uso ‘Refinamento do Casos de Uso <u>Emite Relatório Técnico</u> ’.....	68
Figura 4.12	- Diagrama de Classes do Modelo de Análise. Fase de Elaboração – Interação #1.....	69
Figura 4.13	- Diagrama de Classes do Modelo de Projeto Fase de Elaboração – Interação #1.....	70
Figura 4.14	- Diagrama de Classes do Modelo de Análise Fase de Elaboração – Interação #2.....	71
Figura 4.15	- Pacote Material_Recebido.....	72
Figura 4.16	- Pacote SS.....	72
Figura 4.17	- Pacote Cliente.....	73
Figura 4.18	- Diagrama de Classes do Modelo de Projeto. Fase de Elaboração – Interação #2.....	74
Figura 4.19	- Diagrama de Gráfico de Estados da Classe ‘SS’.....	75
Figura 4.20	- Diagrama de Atividades ‘PRIE’.....	76
Figura 4.21	Diagrama de Atividades ‘Abre SS’.....	77
Figura 4.22	Diagrama de Atividades ‘Conclui SS’.....	79
Figura 4.23	Diagrama de Atividades ‘Emite Relatório Técnico’.....	80

Figura 4.24	Diagrama de Seqüências para o Caso de Uso 'Coleta Informações do Cliente'.....	82
Figura 4.25	Diagrama de Seqüências para o Caso de Uso 'Emite o documento Contrato para a Execução do Serviço'.....	83
Figura 4.26	Diagrama de Seqüências para o Caso de Uso 'Abre SS para o Item Recebido'.....	84
Figura 4.27	Diagrama de Seqüências para o Caso de Uso 'Conclui SS'.....	85
Figura 4.28	Modelo Físico do Banco de Dados.....	86
Figura 4.29	Diagrama de Objetos para o Modelo Físico do Banco de Dados.....	87
Figura 4.30	Tela Principal – Subsistema SGSI.....	91
Figura 4.31	Interface Gráfica para Recepção, identificação e manuseio de Itens de Ensaio.....	92
Figura 4.32	Interface Gráfica para Solicitação de Serviço Cadastro das informações administrativas do serviço.....	93
Figura 4.33	Interface Gráfica para Solicitação de Serviço Cadastro das informações técnicas do serviço.....	94
Figura 4.34	Relatório da 'SS'.....	95
Figura 4.35	Cadastro das informações de devolução do Item de Ensaio.....	96

SUMÁRIO

LISTA DE ABREVIACÕES.....	vii
LISTA DE FIGURAS.....	viii
RESUMO.....	xiv
CAPÍTULO 1 – INTRODUÇÃO.....	1
1.1 – Cenário Atual.....	1
1.2 – Estrutura do Trabalho.....	3
CAPÍTULO 2 – LINGUAGEM DE MODELAGEM UNIFICADA (UML).....	4
2.1 – Itens.....	4
2.1.1 – Itens Estruturais.....	4
2.1.1.1 – Classes.....	5
2.1.1.2 – Objetos.....	5
2.1.1.3 – Casos de Uso.....	6
2.1.2 – Itens Comportamentais.....	6
2.1.2.1 – Interações.....	7
2.1.2.2 – Máquina de Estados.....	7
2.2 – Relacionamentos.....	8
2.2.1 – Dependência.....	8
2.2.2 – Generalização.....	10
2.2.3 – Associação.....	11
2.3 – Diagramas na UML.....	12
2.3.1 – Diagramas Estruturais.....	13
2.3.1.1 – Diagrama de Classes.....	13
2.3.1.2 – Diagrama de Objetos.....	14
2.3.2 – Diagramas Comportamentais.....	16
2.3.2.1 – Diagrama de Casos de uso.....	16
2.3.2.2 – Diagrama de Seqüências.....	17
2.3.2.3 – Diagrama de Atividades.....	19
2.3.2.4 – Diagrama de Gráfico de Estados.....	23
CAPÍTULO 3 – PROCESSO UNIFICADO DE DESENVOLVIMENTO DE SISTEMAS.....	25
3.1 – Características.....	26

3.1.1 – Processo Orientado por Casos de Uso.....	26
3.1.2 – Processo Centrado na Arquitetura.....	27
3.1.3 – Processo Iterativo e Incremental.....	28
3.2 – O Ciclo de Vida.....	30
3.2.1 – Iterações.....	31
3.2.2 – Fluxos de Trabalho.....	32
3.2.2.1 – Requisitos.....	32
3.2.2.2 – Análise.....	34
3.2.2.3 – Projeto.....	36
3.2.2.4 – Implementação.....	38
3.2.2.5 – Teste.....	39
3.2.3 – Fases.....	40
3.2.3.1 – Concepção.....	41
3.2.3.1.1 – Requisitos.....	42
3.2.3.1.2 – Análise.....	42
3.2.3.1.3 – Projeto.....	42
3.2.3.1.4 – Implementação e Teste.....	43
3.2.3.2 – Elaboração.....	43
3.2.3.2.1 – Requisitos.....	44
3.2.3.2.2 – Análise.....	44
3.2.3.2.3 – Projeto.....	45
3.2.3.2.4 – Implementação.....	46
3.2.3.2.5 – Teste.....	46
3.2.3.3 – Construção.....	47
3.2.3.3.1 – Requisitos.....	47
3.2.3.3.2 – Análise.....	48
3.2.3.3.3 – Projeto.....	48
3.2.3.3.4 – Implementação.....	48
3.2.3.3.5 – Teste.....	49
3.2.3.4 – Transição.....	49
3.2.3.4.1 – Atividades da fase de Transição.....	50
CAPÍTULO 4 – ESTUDO DE CASOS.....	52
4.1 – O Laboratório Central da Eletronorte, Lacen.....	52

4.2 – O Sistema Informatizado de Gestão do Laboratório Central, SIGLacen – Descrição do Sistema.....	53
4.2.1 – O Sistema de Solicitação de Serviços (SS).....	56
4.2.2 – A Solicitação de Serviços.....	57
4.3 – O Estudo de Casos propriamente dito.....	58
4.3.1 – Fase de Concepção.....	58
4.3.1.1 – Iteração #1.....	58
4.3.1.1.1 – Fluxo de Requisitos.....	58
4.3.1.1.2 – Fluxo de Análise.....	62
4.3.1.1.3 – Fluxo de Projeto.....	63
4.3.2 – Fase de Elaboração.....	65
4.3.2.1 – Iteração #1.....	65
4.3.2.1.1 – Fluxo de Requisitos.....	65
4.3.2.1.2 – Fluxo de Análise.....	68
4.3.2.1.3 – Fluxo de Projeto.....	69
4.3.2.2 – Iteração #2.....	71
4.3.2.2.1 – Fluxo de Requisitos.....	71
4.3.2.2.2 – Fluxo de Análise.....	71
4.3.2.2.3 – Fluxo de Projeto.....	73
4.3.3 – Fase de Construção.....	81
4.3.3.2.1 – Fluxo de Requisitos.....	81
4.3.3.2.2 – Fluxo de Análise.....	81
4.3.3.2.3 – Fluxo de Projeto.....	81
4.3.3.2.4 – Fluxo de Implementação.....	90
CAPÍTULO 5 – CONCLUSÃO.....	97
REFERÊNCIAS BIBLIOGRÁFICAS.....	99

RESUMO

Este trabalho faz uma abordagem da Linguagem de Modelagem Unificada e a metodologia orientada a objetos do Processo Unificado, dando ênfase aos fluxos de análise e projeto sistemas. Os conceitos adquiridos a partir destas abordagens são aplicados em um estudo de casos do Sistema Informatizado de Gestão do Laboratório Central da Eletronorte, o SIGLacen.

Capítulo 1 – Introdução

1.1- Cenário Atual

A importância da modelagem dentro do desenvolvimento de um sistema é indiscutível.

“A modelagem é a parte central de todas as atividades que levam à implantação de um bom sistema” [1].

Somente com o auxílio da modelagem podemos visualizar e controlar o desenvolvimento de sistemas de maneira eficaz, identificando e gerenciando riscos, estipulando e cumprindo prazos, dentro das estimativas de custo.

Partindo deste princípio várias metodologias para o desenvolvimento de sistemas foram criadas. As primeiras metodologias, classificadas como metodologias estruturadas, caracterizam o desenvolvimento de sistemas em torno de procedimentos e funções.

Muitos sistemas ainda hoje são desenvolvidos com base em metodologias estruturadas, o que os torna instáveis, pelo fato de que a medida em que requisitos se modificam (o que acontece com muita frequência) e o sistema cresce, o trabalho de manutenção do mesmo se torna cada vez mais difícil.

Com a evolução das metodologias, a visão de desenvolvimento de sistemas passou a adotar uma perspectiva diferente. Surgiram então as metodologias orientadas a objetos, que caracterizam o desenvolvimento de sistemas em torno de classes e objetos.

Estas metodologias tiveram grande aceitação na comunidade de desenvolvedores, tendo em vista que o método orientado a objetos possibilitava a construção de sistemas em todos os tipos de domínios de problemas, abrangendo todos os graus de tamanho e complexibilidade.

O sucesso dos métodos orientados a objetos foi fator primordial para que o número de metodologias criadas sob essa perspectiva crescesse de maneira desordenada em um curto espaço de tempo. Inúmeros métodos surgiram, cada um com suas peculiaridades e falhas, e sem nenhum relacionamento com os demais. Desta forma, os usuários destes métodos

sentiram dificuldades em encontrar uma linguagem de modelagem capaz de atender inteiramente às suas necessidades, o que ocasionou a chamada guerra de métodos.

Neste período, os métodos de maior destaque foram o Booch de Grady Booch, OOSE (Engenharia de Software Orientada a Objetos) de Ivar Jacobson, e OMT (Técnica de Modelagem de Objetos) de James Rumbaugh. Estes mesmos autores se sentiram motivados a criar, juntos, uma linguagem unificada de modelagem, pelo fato de seus métodos estarem evoluindo um em direção ao outro de maneira independente. Outro motivo foi o fato de que a unificação de métodos traria estabilidade ao mercado orientado a objetos, permitindo que os projetos tivessem como base uma linguagem madura de modelagem. Além disso, o surgimento de uma linguagem unificada seria um aprimoramento dos métodos já existentes, sendo capaz de solucionar problemas que nenhum método conseguira até então.

Desta forma, através do empenho de Jacobson, Booch e Rumbaugh, surgiu a Linguagem de Modelagem Unificada, a UML.

Porém, a UML é apenas parte de uma metodologia, uma linguagem de modelagem de sistemas, ela apenas disponibiliza as ferramentas necessárias para se criar e ler modelos, mas não aponta quais modelos deverão ser criados, nem quando deverão ser criados. Essa tarefa é de responsabilidade de um processo de desenvolvimento de sistemas.

Mesmo sendo independente de processo, a UML necessitava interagir com uma metodologia específica que pudesse obter o máximo proveito dos recursos da Linguagem de Modelagem Unificada, foi criado então o Processo Unificado.

Com a integração do Processo Unificado à UML respostas para *quem* está fazendo *o que*, *quando* e *como* puderam ser definidas e um padrão no desenvolvimento de sistemas orientados a objetos pode ser estabelecido.

1.2- Estrutura do Trabalho

Este trabalho está dividido em 5 capítulos.

O capítulo 2 faz uma abordagem inicial da UML, destacando os itens e diagramas comportamentais e estruturais que fazem parte da Linguagem de Modelagem Unificada, esta abordagem é de grande relevância para a plena compreensão do Processo Unificado, descrito no capítulo 3 deste trabalho.

A abordagem do capítulo 3 sobre o Processo Unificado envolve a definição de suas características principais, fases e fluxos de trabalho.

Procurando aplicar de maneira prática os conceitos de UML e do Processo Unificado destacando os fluxos de análise e projeto, o capítulo 4 apresenta o estudo de casos do Sistema Informatizado de Gestão do Laboratório Central da Eletronorte, o SIGLacen.

Por fim, o capítulo 5 consiste na conclusão do trabalho.

Capítulo 2 – Linguagem de Modelagem Unificada (UML)

A UML é uma linguagem-padrão para a estruturação de projetos de software. Sua abrangência vai desde a modelagem de sistemas de informação corporativos a serem distribuídos a aplicações baseadas em Web, até sistemas complexos embutidos de tempo real.

Para cumprir seu objetivo, a UML permite que seus usuários modelem um sistema sob diferentes perspectivas. Cada uma destas perspectivas é uma abstração apresentada por diagramas criados a partir dos recursos oferecidos pela linguagem de modelagem.

Em UML, a criação destes diagramas envolve a identificação de itens que formam o vocabulário do sistema e a especificação de como estes itens relacionam-se entre si. Em suma, um diagrama em UML é um conjunto de itens e relacionamentos.

Neste capítulo, será feita, primeiramente, uma abordagem dos itens e relacionamentos da UML, antes de se chegar aos diagramas fornecidos pela linguagem de modelagem unificada.

2.1 - Itens

Dois tipos de itens foram utilizados para compor os diagramas criados para modelar o sistema proposto.

Itens estruturais e itens comportamentais.

2.1.1 - Itens Estruturais

São as partes mais estáticas do modelo e representam elementos conceituais ou físicos.

São itens estruturais:

2.1.1.1 - Classes

“Uma classe é uma descrição de um conjunto de objetos que compartilham os mesmos atributos, operações, relacionamentos e semântica” [1].

Classes são utilizadas para compor o vocabulário do sistema que está sendo desenvolvido através da abstração de objetos que constituem o domínio do problema.

São representadas graficamente por um retângulo dividido em três compartimentos. O primeiro deles contém o nome da classe, que é seu identificador e deve ser escolhido de forma a abranger conceitualmente todos os objetos correspondentes a esta classe. Os segundo e terceiro compartimentos, compreendem respectivamente os atributos e operações da classe em questão, conforme mostra a figura 2.1. Estes atributos e operações devem ser comuns a todos os objetos, instâncias desta classe.

Uma classe obrigatoriamente terá que conter um nome, mas não necessariamente deverá possuir atributos ou operações.

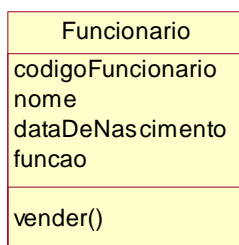


Figura 2.1 - Classe Funcionario.

2.1.1.2 - Objetos

Um objeto ou instância é uma ocorrência de uma classe, ou seja, um objeto possui estado e comportamento específicos e uma identidade única dentro do contexto de uma classe.

Em UML, um objeto é representado semelhantemente a uma classe, porém possui características peculiares como podemos observar na figura 2.2.

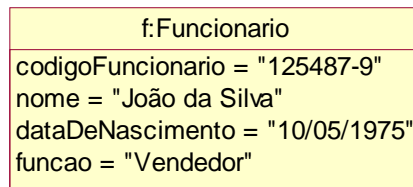


Figura 2.2 – Instância da classe Funcionário.

2.1.1.3 - Casos de Uso

Um caso de uso representa um conjunto de ações realizadas pelo sistema que geram resultados observáveis por um ator, ou seja, um caso de uso é utilizado para estruturar o comportamento de um sistema sem ser necessário especificar sua implementação, além de envolver a interação de atores, que podem ser tanto humanos como sistemas automatizados.

Graficamente, um caso de uso é representado por uma elipse com linhas contínuas, geralmente incluindo somente seu nome, como mostra a figura 2.3.



Solicita Produto

Figura 2.3 – Casos de Uso.

2.1.2 - Itens Comportamentais

Os itens comportamentais são as partes dinâmicas de um modelo em UML e representam comportamentos no tempo e espaço.

Existem dois tipos de itens comportamentais. São eles:

2.1.2.1 - Interações

Uma interação representa um conjunto de mensagens que são trocadas por um conjunto de objetos de um sistema, apresentando, desta forma, o comportamento destes objetos sob um contexto específico.

A modelagem de uma interação pode ser feita dando-se ênfase à ordem temporal das mensagens ou dando-se ênfase a seqüência das mensagens no contexto de alguma organização estrutural de objetos

Uma mensagem é representada graficamente por uma seta cheia, que geralmente inclui o nome de suas operações, como pode ser observado na figura 2.4.

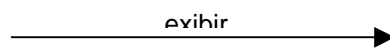


Figura 2.4 – Mensagens.

2.1.2.2 - Máquina de Estados

Uma máquina de estados representa a seqüência de estados que um objeto assume durante o seu tempo de vida, em resposta a eventos. Desta forma, uma máquina de estados modela o comportamento de um objeto individual.

Um estado é uma condição ou situação na vida de um objeto durante a qual ele satisfaz alguma condição, executa alguma atividade ou aguarda um evento.

A modelagem de uma máquina de estados pode ser feita dando-se ênfase ao fluxo de controle de uma atividade para outra, através do diagrama de atividades; ou dando-se ênfase aos estados potenciais dos objetos e às transições entre esses estados, através do diagrama de estados.

Graficamente, um estado é representado como um retângulo com cantos arredondados, geralmente incluindo seu nome e, quando necessário, subestados, conforme mostram, respectivamente, as figuras 2.5 e 2.6.

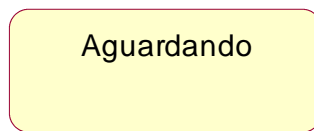


Figura2.5. – Estado.

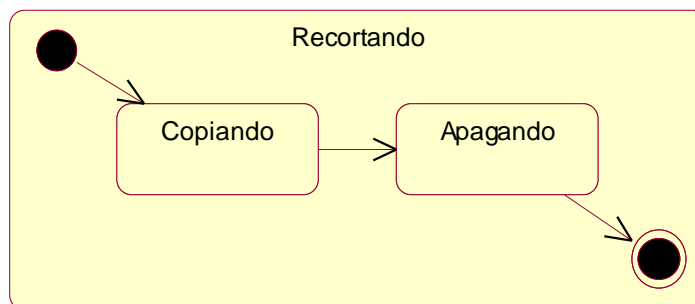


Figura2.6. – Estado e seus subestados.

2.2 - Relacionamentos

Os relacionamentos têm a função de mostrar como os itens abstraídos de um sistema se combinam.

São três os principais tipos de relacionamentos em UML:

2.2.1 - Dependência

É a relação existente entre dois itens quando um dos itens depende do outro, ou seja, na relação de dependência existe um item dependente e outro independente. Desta forma, uma alteração feita no item independente pode afetar a semântica do item dependente.

Uma dependência é representada graficamente por uma seta aberta, com linha tracejada, conforme mostra a figura 2.7.

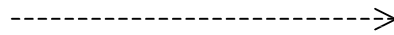


Figura 2.7 – Dependência.

A seta que representa a relação de dependência sai do item dependente e aponta para o item independente, conforme mostra a figura 2.8.

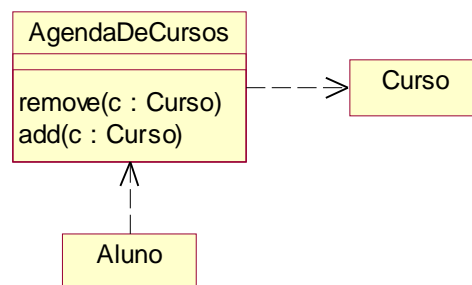


Figura 2.8 – Dependências entre classes.

Analisando a figura 2.8 pode-se perceber dois tipos diferentes de dependências. A mais comum delas acontece quando uma classe só utiliza a outra classe da relação como parâmetro para suas operações, conforme pode ser verificado na relação de dependência entre *AgendaDeCursos* e *Curso*. Normalmente, não se faz necessário mostrar a dependência quando as assinaturas das operações da classe dependente estão definidas por completo, pois os parâmetros das operações mostram a ligação existente com a classe independente de forma explícita. Entretanto, quando as operações estão ocultas ou o modelo apresenta outros relacionamentos além da dependência entre classes, se faz necessário à representação gráfica da dependência.

O outro tipo de dependência é utilizado quando o relacionamento é unilateral, ou seja, a classe independente interage com a classe dependente, que por sua vez, não tem conhecimento da classe dependente. Por exemplo, de acordo com a figura 2.8, a classe *Aluno* interage com a classe *AgendaDeCursos* que, por sua vez, não tem conhecimento da primeira.

2.2.2 - Generalização

Uma generalização é um relacionamento hierárquico entre classes. A classe de maior nível hierárquico é conhecida como superclasse, enquanto a outra classe da relação é conhecida como subclasse.

Graficamente, uma generalização é representada por uma linha contínua com um uma seta em branco apontando sempre a superclasse, conforme mostra a figura 2.9.

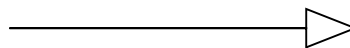


Figura 2.9 – Generalização.

A generalização também é conhecida como um relacionamento “é um tipo de”, ou seja, permite a comparação de duas classes sob a perspectiva de que a subclasse envolvida no relacionamento é um tipo da superclasse relacionada. Um exemplo é o relacionamento entre as classes da figura 2.10.

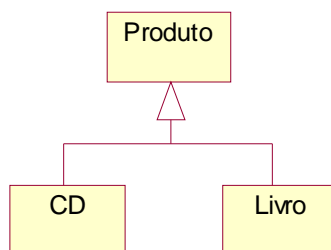


Figura 2.10 – Generalização entre classes.

Analisando a figura 2.10, podemos identificar *CD* e *Livro* como subclasses da superclasse *Produto*, ou seja, podemos afirmar através desse relacionamento hierárquico que a classe *CD* “é um tipo de” *Produto*, assim como *Livro*.

2.2.3 - Associação

Uma associação é um relacionamento estrutural que conecta classes, ou seja, através de uma associação entre classes, pode-se chegar em um objeto de uma classe a partir do objeto da outra classe relacionada.

A representação desta relação estrutural entre objetos de classes relacionadas por uma associação, pode ser adornada por um nome, um papel e a multiplicidade do relacionamento.

O nome identifica a associação entre as classes e uma seta auxiliar pode ser utilizada para indicar a direção correta de leitura da associação.

Um papel pode ser mencionado quando apenas um conjunto de objetos de uma classe atende os requisitos estruturais de uma associação, ou quando for necessário mencionar qual papel determinada classe assume diante uma associação com outra classe.

A multiplicidade, por sua vez, é a representação da quantidade de objetos envolvidos em uma associação com relação aos objetos da outra classe. Por exemplo, a multiplicidade existente na extremidade A de uma associação especifica que para cada objeto da classe da extremidade B, deve existir a mesma quantidade de objetos especificados na classe da extremidade A.

Graficamente, uma associação é representada por uma linha contínua e adornos, conforme mostra a figura 2.11.

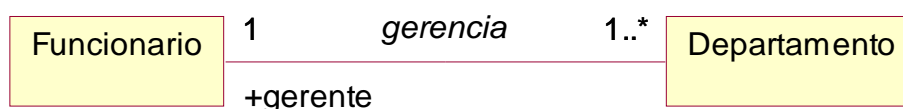


Figura 2.11 – Associação.

Um tipo especial de associação é a agregação. Este tipo específico de associação é utilizado quando se deseja criar um relacionamento “todo/parte”, onde uma classe representa um item maior (o todo), formado por itens menores (as partes). A agregação também é conhecida como relacionamento

“tem-um”, ou seja, dizemos que um objeto gerado da classe “todo” tem um objeto gerado da classe “parte”.

Graficamente uma agregação é representada por uma linha contínua com um losango em uma das extremidades, sendo que esta extremidade está conectada com a classe “todo”, conforme mostra a figura 2.12.



Figura 2.12 – Agregação.

Analisando a figura 2.12, percebe-se que uma agregação também pode possuir adornos para especificar mais detalhadamente o relacionamento todo/parte. Neste caso, podemos afirmar que uma Loja tem um ou vários Departamentos.

Outra forma de se representar uma associação é utilizando atributos de ligação, também conhecidos como classes de associação. Isto é feito quando uma associação entre duas classes possui propriedades tão específicas que precisam ser modeladas em uma terceira classe. Por exemplo, em um relacionamento entre as classes *Funcionário* e *Produto* existe a associação de *venda*, esta associação possui suas propriedades específicas como data da venda e desconto. Conforme mostra a figura 2.13.

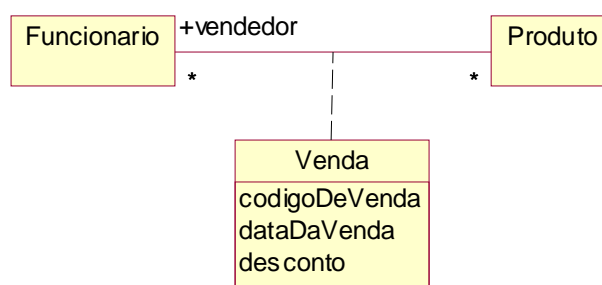


Figura 2.13 – Atributo de Ligação ou Classe de Associação.

2.3 - Diagramas na UML

Diagramas são recursos gráficos utilizados para se visualizar um sistema sob diferentes perspectivas e são geralmente compostos por itens e

relacionamentos. A UML oferece nove tipos diferentes de diagramas que podem ser classificados como Diagramas Estruturais ou Diagramas Comportamentais. Os principais diagramas da UML são apresentados nesta abordagem, em particular, aqueles utilizados no estudo de casos do sistema proposto apresentado no capítulo 4.

2.3.1- Diagramas Estruturais

Em UML, os aspectos estáticos de um sistema, ou seja, a representação de seu esqueleto e estrutura, relativamente estáveis, são visualizados através dos diagramas estruturais.

2.3.1.1 - Diagrama de Classes

São os diagramas mais utilizados em sistemas de modelagem orientados a objetos. O diagrama de classes é composto basicamente por um conjunto de classes relacionadas entre si.

Dizemos que um diagrama de classes modela os aspectos estáticos de um sistema pelo fato de sua estrutura ser sempre válida em qualquer ponto do ciclo de vida do sistema.

Um diagrama de classes pode ser utilizado para três propósitos básicos e distintos, são eles:

1 - Fazer a modelagem do vocabulário de um sistema

Neste caso, o diagrama de classes é utilizado para indicar quais abstrações fazem parte do sistema e quais estão fora do limite do mesmo.

2 - Fazer a modelagem de colaborações simples

Ao modelar colaborações simples o diagrama de classes representa um conjunto de classes, interfaces e outros elementos

que funcionam em conjunto para proporcionar algum comportamento cooperativo, ou seja, o diagrama mostra como as classes trabalham em conjunto permitindo a compreensão de uma semântica maior do que se estas mesmas classes fossem analisadas individualmente.

3 - Fazer a modelagem do esquema lógico de um banco de dados

Utilizado para fazer a modelagem orientada a objetos do esquema lógico de um banco de dados físico que pode ser orientado a objetos, relacional ou híbrido.

A figura 2.14 mostra um diagrama de classes para uma loja de cd's e livros.

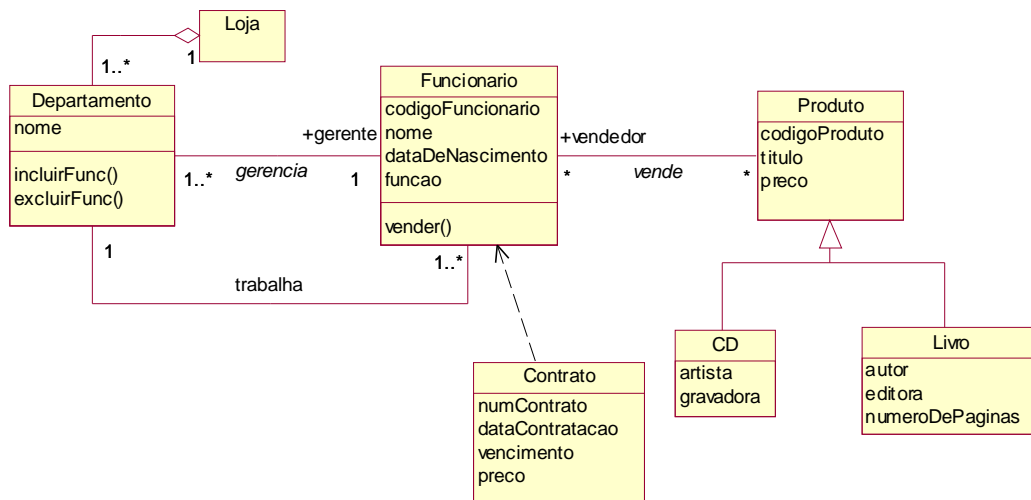


Figura 2.14 – Diagrama de Classes.

2.3.1.2 – Diagrama de Objetos

O diagrama de objetos nada mais é que um diagrama de classes instanciado, ou seja, o diagrama de objetos modela instâncias dos itens de um diagrama de classes e seus respectivos relacionamentos. Desta forma, os itens do diagrama de objetos assumem valores, correspondentes a seus atributos,

que no seu conjunto representam a situação estática do sistema em um determinado ponto no tempo.

A modelagem deste diagrama é feita considerando apenas um conjunto de objetos possíveis para o sistema em um determinado instante, tendo em vista que é praticamente impossível modelar todos os objetos que um sistema pode assumir.

O diagrama de objetos é utilizado para fazer a modelagem da visão de projeto estática ou da visão de processo estática de um sistema. A primeira perspectiva, ou seja, a visão de projeto de um sistema, proporciona o suporte necessário para visualizar os serviços que o sistema deverá fornecer a seus usuários finais; a segunda, por sua vez, refere-se às questões de concorrência, desempenho e escalabilidade. Em qualquer um dos dois casos o diagrama de objetos assume o papel de modelador de estruturas dos objetos, que em conjunto, expressam determinada semântica contida no sistema em questão.

A figura 2.15 mostra um diagrama de objetos construído com base no diagrama de classes apresentado na figura 2.14.

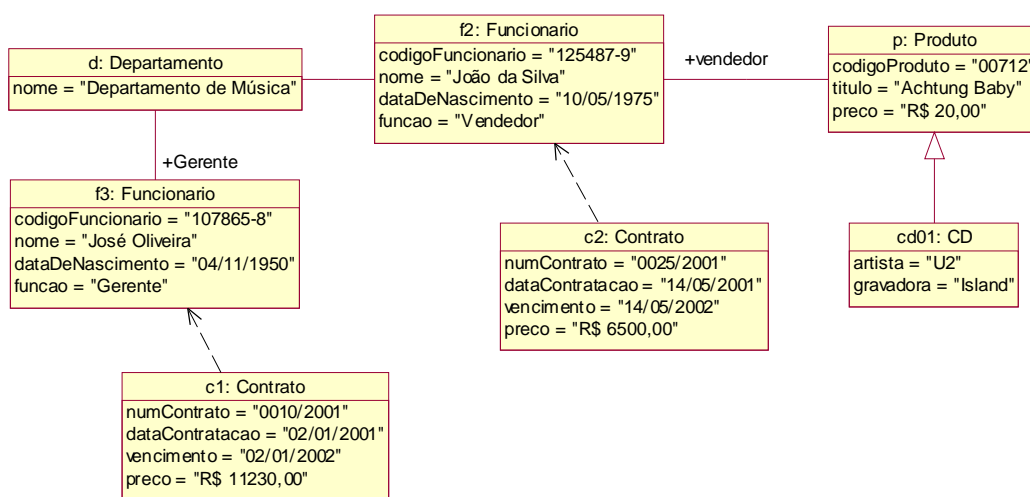


Figura 2.15 – Diagrama de Objetos.

2.3.2- Diagramas Comportamentais

Os diagramas comportamentais são responsáveis pela modelagem dos aspectos dinâmicos de um sistema, ou seja, pela representação das partes do sistema que sofrem alterações, como por exemplo, o fluxo de mensagens ao longo do tempo e a transição de estados que o mesmo pode assumir.

2.3.2.1 - Diagrama de Casos de Uso

Os diagramas de casos de uso assumem papel importante na modelagem comportamental de um sistema, um subsistema ou uma classe, por apresentar uma visão externa de como esses elementos podem ser utilizados no contexto.

Os elementos que compõe um diagrama de casos de uso são os próprios casos de uso, os atores e os relacionamentos existentes entre esses dois primeiros elementos.

Um caso de uso especifica o comportamento de um sistema ou de parte dele através de um conjunto de seqüências de ações executadas dentro do contexto deste sistema e que produz um resultado de valor observável por um ator, este por sua vez, é o elemento externo que interage com o sistema, que pode ser um usuário, um hardware ou até mesmo outro sistema.

Dentro da modelagem da arquitetura de um sistema, a visão de caso de uso abrange os casos de uso que descrevem o comportamento do sistema conforme visto pelos seus usuários finais, analistas e pessoal de teste. Os aspectos estáticos desta visão são capturados em diagramas de caso de usos.

Um diagrama de casos de uso pode ser utilizado para dois propósitos básicos e distintos, são eles:

1 – *Fazer a modelagem do contexto de um sistema:*

Esta modelagem tem como principal função especificar quais atores estão de fora do contexto do sistema, como eles interagem com o mesmo, e seus respectivos papéis.

2 – Fazer a modelagem dos requisitos de um sistema:

Esta modelagem especifica o que o sistema deverá fazer sob um ponto de vista externo, ou seja, a modelagem de requisitos, utilizando o diagrama de casos de uso, permitirá visualizar os fatores externos que interagem com o sistema e como este sistema reage a estas interações, mas sem especificar o funcionamento interno do mesmo.

A figura 2.16 mostra um diagrama de casos de uso baseado no diagrama de classes apresentado na figura 2.14.

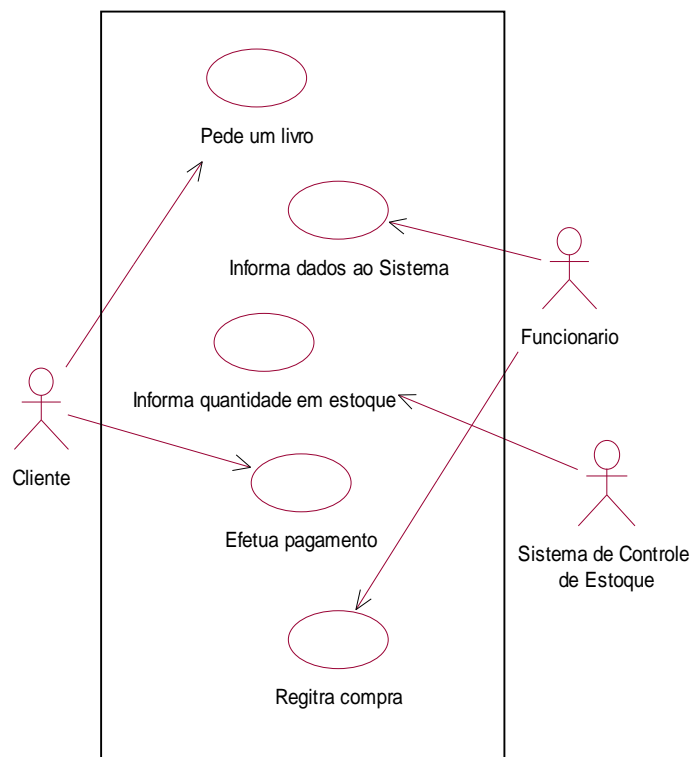


Figura 2.16 – Diagrama de Casos de Uso.

2.3.2.2 - Diagrama de Seqüências

O diagrama de seqüências é um diagrama de interação responsável pela modelagem dos aspectos dinâmicos de um sistema e que dá ênfase à ordenação temporal das mensagens trocadas entre objetos deste sistema.

Desta forma, para proporcionar a melhor visualização do fluxo de controle ao longo do tempo, o diagrama de seqüências é organizado colocando-se os objetos correspondentes na parte superior do diagrama, ao longo do eixo horizontal; e suas respectivas mensagens são colocadas ao longo do eixo vertical, em uma ordem cronológica, de cima para baixo. As mensagens são representadas por setas horizontais, onde a ponta da seta identifica o objeto alvo.

Também existem no diagrama de seqüências as linhas de vida dos objetos que são as linhas verticais tracejadas que representam o tempo de existência de um objeto. Outro item é o foco de controle, um retângulo alto e estreito, que mostra o período durante o qual um objeto está desempenhando determinada ação.

Um diagrama de seqüências faz a modelagem dinâmica de um sistema sob diferentes contextos. Para modelar o comportamento do sistema sob diferentes níveis de abstração, o diagrama de sistemas é criado com um contexto específico para interação, que pode ser um sistema como um todo, um subsistema, uma operação ou uma classe.

Desta forma, um diagrama de seqüências ao ser criado, deve ser associado a um diagrama de casos de uso ou a um diagrama de classes ou objetos.

O diagrama de seqüências apresentado na figura 2.17 refere-se ao diagrama de casos de uso da figura 2.16.

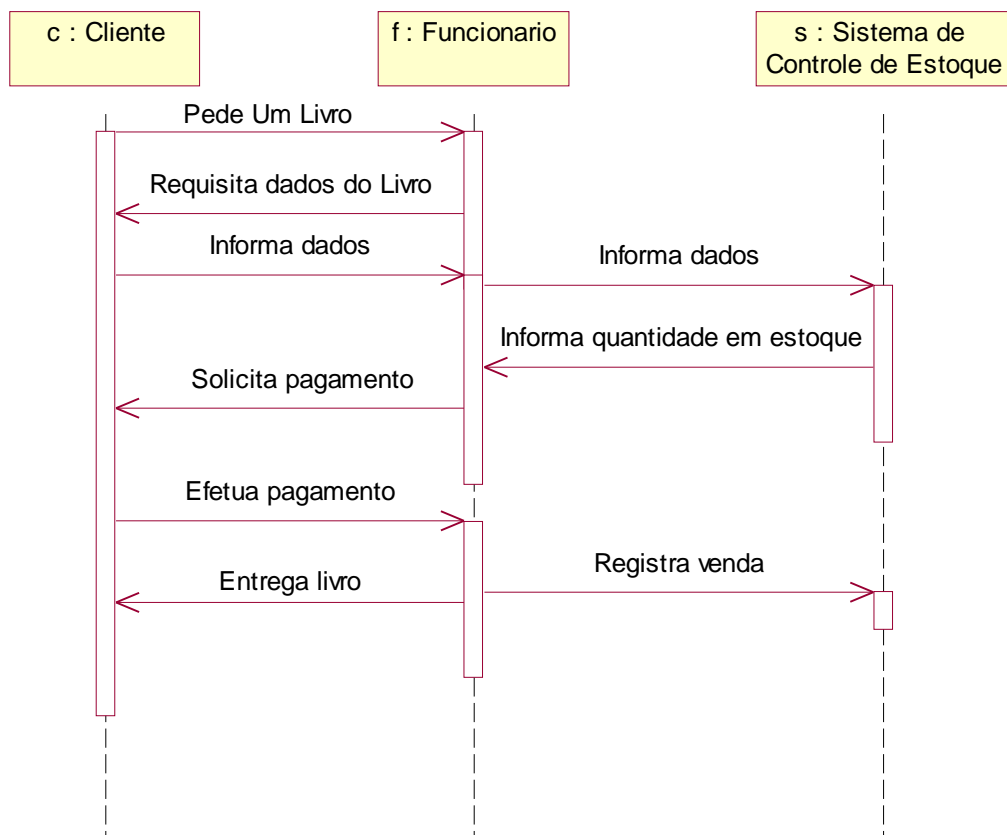


Figura 2.17 – Diagrama de Seqüências.

2.3.2.3 - Diagrama de Atividades

O diagrama de atividades é responsável pela modelagem do fluxo de controle entre as atividades de um sistema, ou seja, ele modela o fluxo seqüencial das atividades geradas por uma operação do sistema ou por um agente externo.

O diagrama de atividades é composto por estados de atividades, estados de ação e transições.

Estados de ação são computações atômicas executáveis, ou seja, computações que não podem ser interrompidas ou decompostas, e que representam um estado do sistema sob a execução de uma ação. Os estados de atividades, por sua vez, são não-atômicos, ou seja, podem ser interrompidos e decompostos para serem representados mais detalhadamente

por outro diagrama de atividades. Em suma, um estado de ação é um estado de atividade que não pode mais ser decomposto. Notacionalmente, não existe diferença entre estados de ação e de atividades.

As transições em um diagrama de atividades podem ser de dois tipos:

1 - *Seqüências simples*

É uma linha simples com uma direção, que liga dois estados, e representa o fluxo de controle entre eles.

2 - *Ramificações*

A ramificação é representada por um losango que indica caminhos alternativos a tomar em um diagrama mediante alguma expressão. A ramificação poderá ter uma transição de entrada e duas ou mais de saída, sendo que apenas uma transição de saída pode ser tomada por vez, obedecendo a expressão relacionada à ramificação correspondente.

Desta forma, as transições são utilizadas para modelar o fluxo de controle de um estado de ação ou de atividade, para outro estado de ação ou de atividade, assim que o primeiro estiver concluído, sendo que a partir da conclusão deste, o fluxo de controle poderá passar imediatamente para outro estado obedecendo ou não a determinada condição. Porém, quando se fizer necessário modelar fluxos de controle concorrentes, o diagrama de atividades deverá utilizar barras de sincronização.

As barras de sincronização são representadas por linhas horizontais ou verticais no diagrama e indicam a união ou a bifurcação de fluxos de controles. A bifurcação poderá ter uma única transição de entrada e duas ou mais de saída, sendo que cada transição de saída representa um fluxo de controle independente e paralelo. A união poderá ter duas ou mais transições de entrada e uma única transição de saída, esta transição de saída indica a sincronização dos fluxos de controle de entrada.

O diagrama de atividades é uma variação do diagrama de gráfico de estados, sendo que o diagrama de atividades não necessita especificar qual

evento causou a mudança de um estado para outro no sistema. Outra característica peculiar do diagrama de atividades é o fato deste ser modelado como um fluxograma, detalhando uma determinada operação de um sistema e também podendo considerar agentes externos que interagem com o mesmo.

A figura 2.18 mostra um diagrama de atividades para o caso de uso *Registra Compra* do diagrama de casos de uso da figura 2.16.

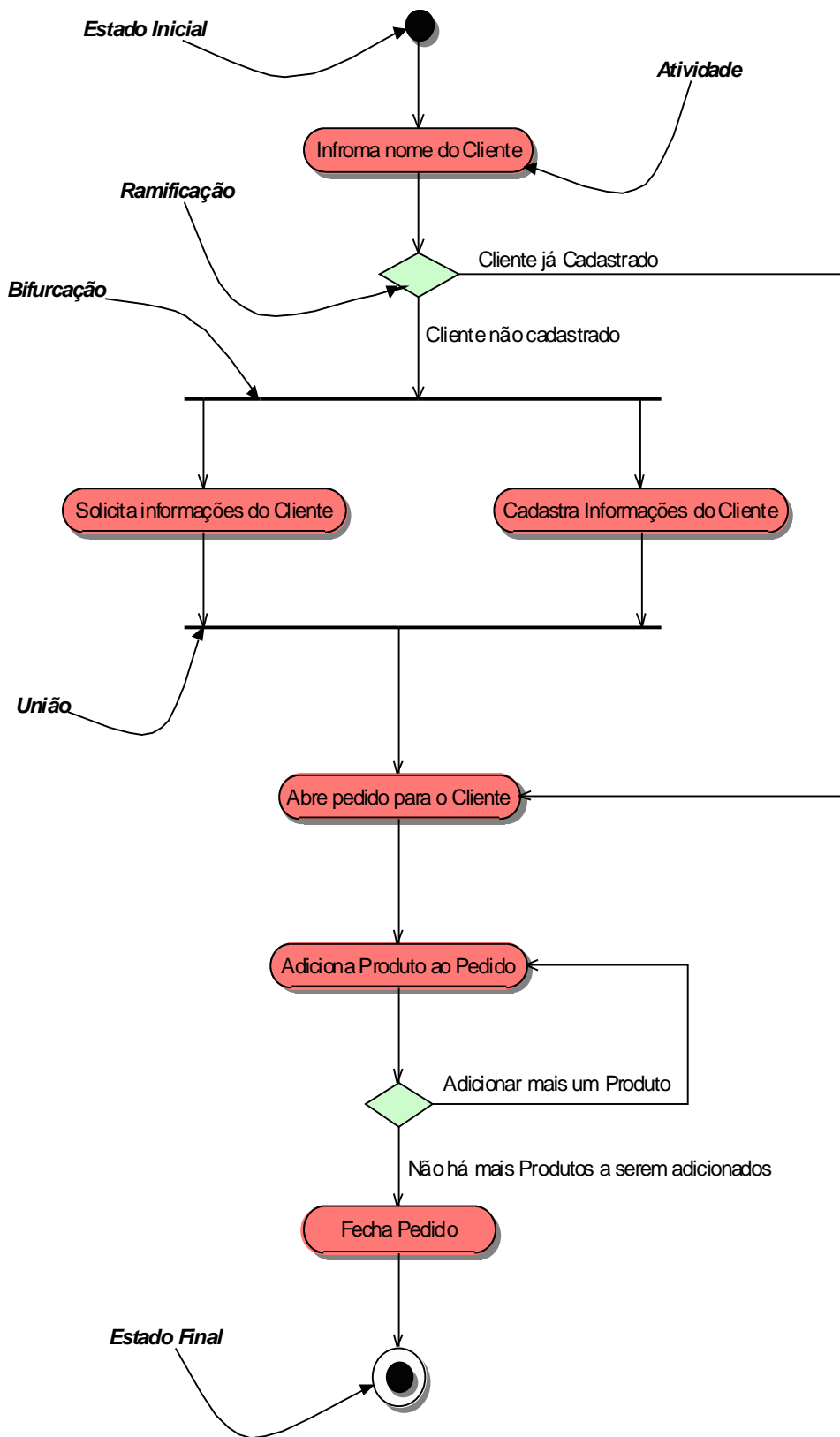


Figura 2.18 – Diagrama de Atividades.

2.3.2.4 - Diagrama de Gráficos de Estados

Ao modelar os aspectos dinâmicos de um sistema o diagrama de gráfico de estados envolve a modelagem comportamental de objetos reativos. Um objeto é dito reativo quando responde a eventos externos ao seu contexto. Enquanto um diagrama de atividades é responsável pela modelagem do fluxo de controle entre atividades, o diagrama de gráfico de estados, por sua vez, é responsável pela modelagem do fluxo de controle entre estados.

O diagrama de gráfico de estados modela este fluxo de controle de um estado para outro através de máquinas de estados.

Uma máquina de estados é o conjunto de estados pelo qual um objeto passa quando responde a um determinado evento. Podemos caracterizar um estado como sendo uma condição ou situação em que um objeto se encontra e um evento como uma ocorrência significativa que, neste caso, é capaz de ativar uma transição de estado. Uma transição, por sua vez, é um relacionamento entre dois estados onde um objeto no primeiro estado realiza determinadas ações e passa para o segundo estado quando um evento é acionado e os requisitos são satisfeitos. Uma atividade é uma execução não-atômica em uma máquina de estados e uma ação é uma execução atômica que resulta em uma alteração de estado ou no retorno de um valor.

Um diagrama de gráfico de estados é composto por estados simples e compostos e transições, incluindo eventos e ações.

A figura 2.19 mostra um diagrama de gráfico de estados referente ao ator *Sistema de Controle de Estoque*, existente no diagrama de casos de uso da figura 2.16.

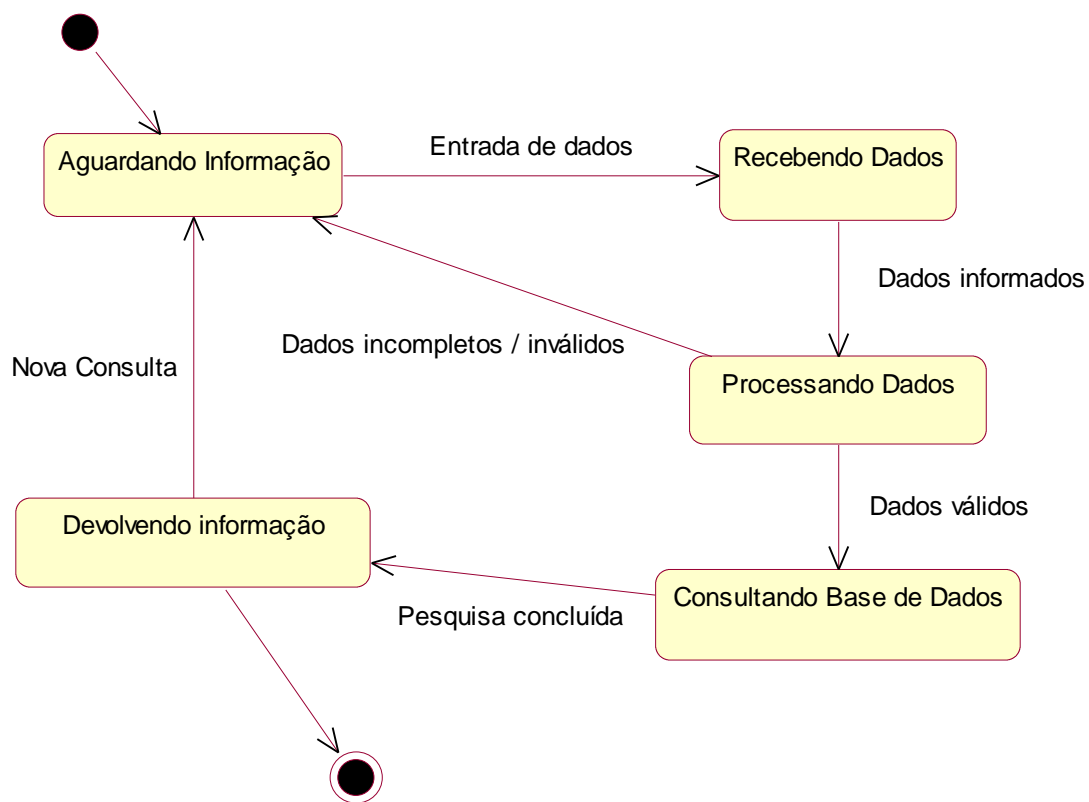


Figura 2.19 – Diagrama de gráfico de Estados.

Capítulo 3 – Processo Unificado de Desenvolvimento de Sistemas

“Um processo é um conjunto de passos que define *quem* está fazendo o *que*, *quando* e *como* para alcançar determinado objetivo” [2]. Na engenharia de software este objetivo é entregar de maneira eficiente e previsível, um produto capaz de atender às necessidades de seu negócio.

A UML por ser apenas uma linguagem para modelagem orientada a objetos e amplamente independente de processo, indica apenas como criar e ler modelos bem-formados, mas não aponta quais modelos serão criados nem quando deverão ser criados. Essa tarefa cabe ao processo de desenvolvimento de sistemas.

A UML é, portanto, apenas parte de um método para o desenvolvimento de sistemas, e o fato de ser independente de processo permite que possa ser utilizada com vários processos de engenharia de software.

Entretanto, a adoção do Processo Unificado, como o processo responsável pelo desenvolvimento de sistemas modelados em UML, é mais adequada, tendo em vista que este processo está diretamente ligado à Linguagem de Modelagem Unificada, utilizando-a como notação de uma série de modelos que compõem os principais resultados das atividades do processo. Além disso, os principais responsáveis pela criação do Processo Unificado foram Grady Booch, James Rumbaugh, e Ivar Jacobson, os mesmos desenvolvedores da UML.

O Processo Unificado captura algumas das melhores práticas atuais de desenvolvimento de software de forma que pode ser adaptada a uma ampla variedade de projetos e empresas. Porém, o Processo Unificado não pode ser considerado como apenas uma união das melhores e principais características das metodologias mais populares, ele também possui características específicas, como ser orientado por casos de uso, ser centrado na arquitetura, ser iterativo e incremental. Estas são as características principais do Processo Unificado, o que o torna único.

Este capítulo fará uma abordagem sobre processo unificado, definindo suas características, assim como seus ciclos de desenvolvimento, as fases que compõem estes ciclos e os fluxos de trabalho das iterações que subdividem as fases do ciclo de vida do desenvolvimento de um software.

3.1 – Características

São três as características que fazem com que o Processo Unificado seja único, ser orientado por casos de uso, ter a arquitetura centrada e ser iterativo e incremental. Estas três características se relacionam e são igualmente importantes. A arquitetura fornece a estrutura que guia o trabalho de cada iteração, os casos de uso definem as metas e dirigem o trabalho de cada iteração. Remover uma das três características reduziria drasticamente o valor do Processo Unificado. Estas três características funcionam como o tripé onde o Processo Unificado é apoiado, se uma das pernas deste tripé for retirada, o processo cai.

3.1.1 – Processo orientado por Casos de Uso

Um caso de uso é uma seqüência de ações de um sistema que devolve ao usuário um resultado de valor. Um conjunto de casos de uso definido sob determinado contexto forma o diagrama de casos de uso que descreve a funcionalidade do sistema sob este contexto. Em outras palavras, um digrama de casos de uso define a funcionalidade de um sistema para cada usuário do mesmo.

Um processo orientado por casos de uso faz com que um sistema seja desenvolvido sob a perspectiva de atender, especificamente, às necessidades de cada usuário que interage com o sistema, evitando, desta forma, que o sistema possa ser desenvolvido a ponto de apresentar funcionalidades desnecessárias. Entretanto, casos de uso não são ferramentas ligadas apenas à especificação de requisitos do sistema, mas também ao projeto, implementação e testes do mesmo, ou seja, o processo de desenvolvimento do sistema é orientado por casos de uso.

Ser orientado por casos de uso significa que o processo de desenvolvimento segue um fluxo, ou seja, o processo passa por uma série de fluxos de trabalho que derivam dos casos de uso.

Desta forma, os modelos de análise, projeto e implementação são criados pelos desenvolvedores com base no modelo de casos de uso. Este processo é conhecido como realização de casos de uso e é evidenciado durante todo o ciclo de vida do Processo Unificado.

Casos de uso são desenvolvidos de acordo com a arquitetura do sistema. Assim, eles definem a arquitetura do sistema, e esta, por sua vez, influencia a seleção dos casos de uso. Conseqüentemente, tanto a arquitetura do sistema quanto os casos de uso, evoluem durante o ciclo de vida do sistema.

3.1.2 – Processo centrado na Arquitetura

“A arquitetura é a visão de todos os modelos que juntos representam o sistema como um todo” [2].

O conceito de arquitetura de software engloba os aspectos estáticos e dinâmicos mais significantes do sistema. A arquitetura cresce além das necessidades do empreendimento, como percebido pelos usuários e suporte, e refletido nos casos de uso.

De qualquer modo, a arquitetura também é influenciada por muitos outros fatores, como a plataforma na qual o software será implantado, os blocos de construção reutilizáveis, requisitos de desenvolvimento e requisitos não-funcionais.

A arquitetura é uma visão do projeto do sistema como um todo, destacando suas características mais importantes, mas sem entrar em detalhes. Considerando que “o que é significativo em um sistema depende do ponto de vista de cada desenvolvedor e que muitas vezes uma opinião sensata esta ligada à experiência, o valor da arquitetura depende das pessoas que estão ligadas ao desenvolvimento do sistema” [2]. De qualquer modo, o

processo ajuda o arquiteto a concentrar-se nas metas corretas, como inteligibilidade, poder de recuperação para mudanças futuras e reutilização.

A relação existente entre casos de uso e a arquitetura é que os casos de uso estão ligados a funcionalidade de um sistema e a arquitetura, por sua vez, está ligada à forma deste. Além disso, funcionalidade e forma devem estar balanceadas para se alcançar um produto final de qualidade, ou seja, casos de uso e arquitetura devem estar ligados a tal ponto que o primeiro seja desenvolvido de acordo com a arquitetura, e esta por sua vez, forneça um ambiente para a realização de todos os requisitos dos casos de uso.

Assim, a arquitetura de um sistema deve ser projetada a ponto de permitir que o sistema evolua, não apenas durante o início de seu desenvolvimento, mas também através de gerações futuras. Para alcançar este objetivo, os arquitetos devem trabalhar com as funções-chaves de um sistema, ou seja, os casos de uso-chaves de um sistema. Estes são apenas cerca de 5% a 10% do total de casos de uso, porém, os mais significativos, aqueles que constituem o núcleo de funções do sistema.

3.1.3 – Processo Iterativo e Incremental

Durante o desenvolvimento de um sistema, é prático dividi-lo em mini-projetos. Cada mini-projeto é uma iteração que resulta em um incremento. Iterações referem-se às etapas do fluxo de trabalho, e incrementos, ao avanço no desenvolvimento do produto. Para serem mais efetivas, as iterações devem ser controladas, ou seja, devem ser executadas de modo planejado. Por isso são consideradas mini-projetos.

Os desenvolvedores baseiam sua escolha, a respeito do que será implementado em uma iteração, em dois fatores. Primeiro, a iteração lida com um grupo de casos de uso que juntos representam o funcionamento do sistema em desenvolvimento. Segundo, a iteração lida com os riscos mais significativos do empreendimento. Iterações sucessivas constroem um conjunto de artefatos a partir do estado em que estes foram deixados ao término da iteração passada. Desta forma, partindo dos casos de uso, o mini-projeto prossegue

através dos fluxos de trabalho subsequentes (análise, projeto, implementação e teste) até alcançar a forma de código executável.

Em cada iteração, os desenvolvedores identificam e especificam os casos de uso relevantes, criam o projeto de acordo com a arquitetura escolhida, implementam o projeto em componentes e verificam se os componentes satisfazem os casos de uso. Se uma iteração alcançar seu objetivo, o que geralmente ocorre, o desenvolvimento prossegue com a próxima iteração. Porém, se uma iteração não alcançar seu objetivo, os desenvolvedores devem revisar as decisões tomadas anteriormente e tentar uma nova abordagem.

Para conseguir maior economia no desenvolvimento, a equipe de projeto tenta selecionar apenas as iterações necessárias para alcançar as metas predefinidas. Um projeto bem sucedido avança por um curso com o mínimo de desvios do curso planejado inicialmente pelos desenvolvedores. Por outro lado, o surgimento de problemas imprevistos acarreta no aumento do número de iterações ou alteração na seqüência das mesmas, fazendo com que o processo de desenvolvimento tome mais tempo e dedicação. Minimizar a possibilidade de surgimento de problemas imprevistos é uma das metas da redução de riscos.

Um processo iterativo controlado tem várias vantagens:

- Iterações controladas reduzem o risco de custo para as despesas em um único incremento. Se os desenvolvedores precisarem repetir a iteração, será perdido apenas o esforço dedicado a uma iteração, não ao produto como um todo.
- Iterações controladas reduzem o risco de violação de prazos. O fato de identificar problemas no início do desenvolvimento, permite que os desenvolvedores aloquem tempo para resolve-los sem extrapolar prazos.
- Iterações controladas aceleram o tempo de desenvolvimento do projeto como um todo, pelo fato de desenvolvedores trabalharem de

maneira mais eficiente em cima de resultados de escopo pequeno e claro.

- Iterações controladas reconhecem uma realidade muitas vezes ignorada, que a necessidade dos usuários e requisitos correspondentes não podem ser completamente definidos no início do desenvolvimento. Eles devem ser refinados em sucessivas iterações. Este modo de operação torna mais fácil a adaptação do sistema a mudanças dos requisitos.

3.2 – O Ciclo de Vida

O processo unificado consiste da repetição de uma série de ciclos durante a vida de um sistema, como mostrado na Figura 3.1.

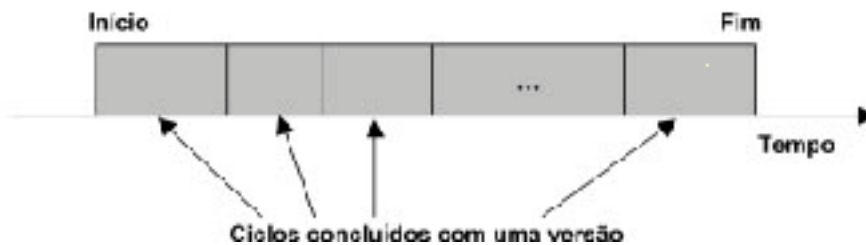
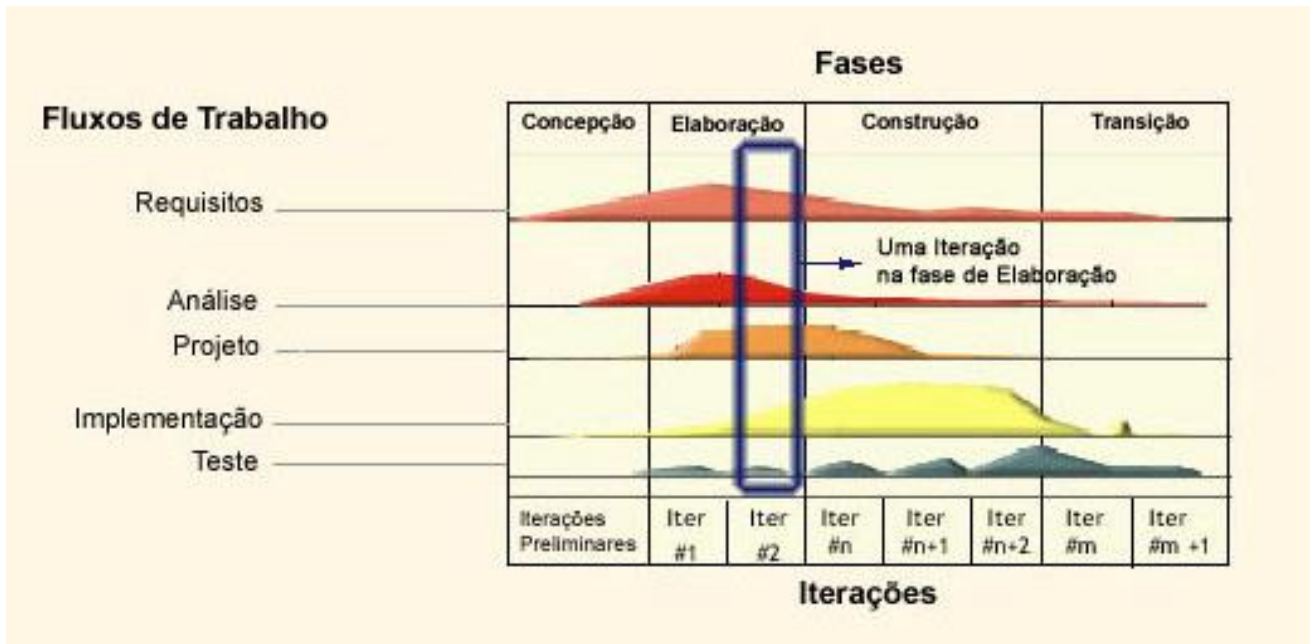


Figura 3.1 – A vida de um processo em ciclos.

Cada ciclo é concluído com uma versão do produto pronta para distribuição e é subdividido em quatro fases, concepção, elaboração, construção e transição. Cada fase, por sua vez, é subdividida em iterações que passam por todos os cinco fluxos de trabalho do processo, requisitos, análise, projeto, implementação e teste.

O ciclo de vida do Processo Unificado é mostrado na figura 3.2.



3.2.1 - Iterações

Antes de sua conclusão, o ciclo de vida do processo unificado passa por várias e sucessivas iterações. Cada uma destas iterações resulta em uma versão de um produto executável que constitui um subconjunto do produto final em desenvolvimento e cresce de modo incremental de uma iteração para outra até se tornar o sistema final, conforme mostra a figura 3.3.

Cada iteração corresponde a uma das quatro fases do ciclo de vida, concepção, elaboração, construção e transição. Em cada uma destas fases, ocorrem várias iterações, sendo que cada iteração passa por todos os cinco fluxos de trabalho do Processo Unificado, requisitos, análise, projeto, implementação e teste. Porém, a importância de cada fluxo de trabalho para uma iteração depende da fase em que esta iteração se encontra.

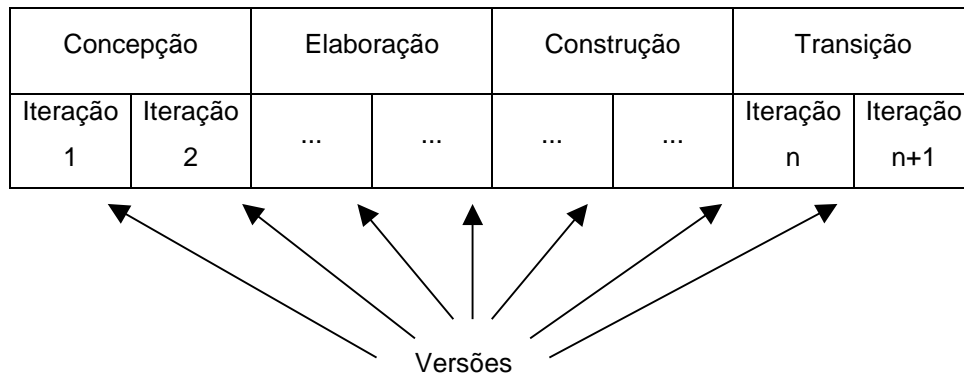


Figura 3.3 – Um ciclo com suas fases e iterações.

Durante a concepção, o foco está na captação de requisitos. Durante a elaboração, o foco passa a ser a análise e projeto do sistema. A implementação é a atividade central na construção e a transição é caracterizada pela entrega de um produto á uma comunidade de usuários.

3.2.2 – Fluxos de Trabalho

Uma iteração típica realiza cinco atividades ou fluxos de trabalho. Porém, a importância de cada fluxo de trabalho depende da fase em que a iteração se encontra. Os cinco fluxos de trabalho do processo unificado são:

3.2.2.1 – Requisitos

Durante este fluxo de trabalho, os requisitos do sistema são especificados através da identificação das necessidades de usuários e clientes. Estes requisitos funcionais são expressos em casos de uso através do modelo de casos de uso. Um modelo de casos de uso é composto por todos os atores e casos de uso de um sistema, ou seja, é composto pelo conjunto de diagramas de casos de uso que compõem o sistema, e especifica como esse

sistema será utilizado sob a perspectiva de clientes, usuários e desenvolvedores.

A identificação de casos de uso é realizada através do estudo de como os usuários precisam usar o sistema para realizar seu trabalho. Qualquer maneira válida de utilização do sistema para um usuário é considerada um caso de uso candidato. Estes candidatos serão, então, elaborados, mudados, divididos em caso de uso menores ou incorporados em casos de uso mais completos.

O modelo de casos de uso é desenvolvido em vários incrementos, onde as iterações irão adicionar novos casos de uso e/ou adicionar detalhes as descrições dos casos de uso existentes. A figura 3.4 mostra a importância do fluxo de requisitos em cada fase do ciclo de vida do sistema.

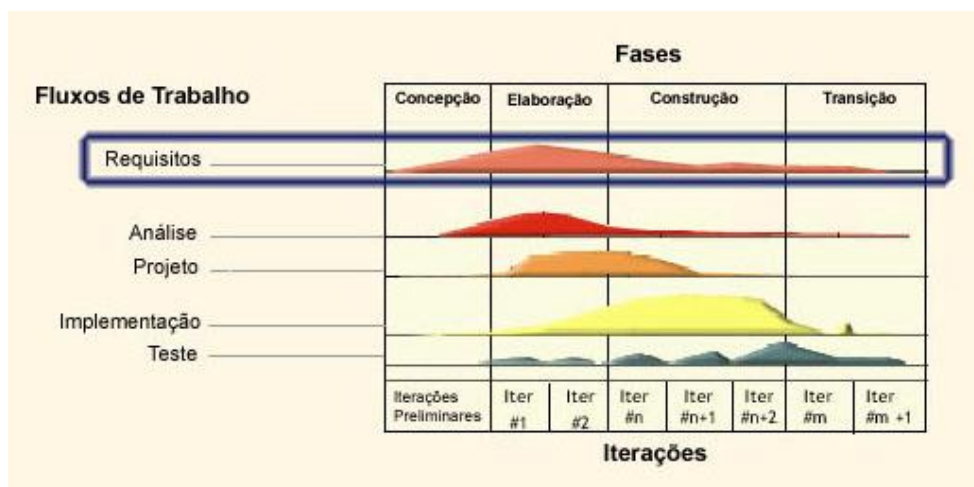


Figura 3.4 – Fluxo de Requisitos.

Durante a fase de concepção, os casos de uso mais importantes são identificados, delimitando o domínio do sistema. Durante a fase de elaboração, a maioria dos requisitos remanescentes é capturada, assim, desenvolvedores poderão saber o quão deverão se empenhar para desenvolver o sistema. Ao final da fase de elaboração, devem ter sido capturados e descritos cerca de 80% dos requisitos do sistema, porém, apenas 5% a 10% destes requisitos são implementados nesta fase. Os requisitos que sobram são capturados e implementados durante a fase de construção. Na fase de transição quase não

há requisitos a serem capturados, a menos que ocorram mudanças nos mesmos.

O modelo de casos de uso estará concluído quando representar todos os requisitos funcionais do sistema de forma que usuários, clientes e desenvolvedores possam entender.

Portanto, o fluxo de trabalho de requisitos foca suas atividades na identificação de entidades que interagem com o sistema (atores) e dos requisitos funcionais deste sistema para cada um dos atores (casos de uso) e no agrupamento destes elementos sob determinado contexto de forma a construir diagramas de casos de uso que no seu conjunto formarão o modelo de casos de uso.

De fato, o modelo de casos de uso é uma ferramenta utilizada para organizar os requisitos em uma forma fácil de gerenciar. Clientes e usuários devem entendê-lo e usá-lo para comunicar suas necessidades de forma consistente e não-redundante. Desenvolvedores podem dividir o trabalho de captura de requisitos entre si e, então, utilizar os resultados como entrada para os fluxos de análise, projeto, implementação e teste.

3.2.2.2 – Análise

O produto do fluxo de análise é o modelo de análise. Este modelo tem a função de refinar os requisitos especificados no fluxo anterior (fluxo de requisitos) através da construção de diagramas de classes conceituais, permitindo, desta forma, argumentações a respeito do funcionamento interno do sistema. Além disso, o modelo de análise fornece mais poder expressivo e formalismo, como diagramas de interações e diagrama de gráficos de estados que representam a dinâmica do sistema.

De fato, o fluxo de análise tem maior importância durante a fase de elaboração, como pode ser visto na figura 3.5. Isso contribui para a definição de uma arquitetura estável e facilita o entendimento detalhado dos requisitos.

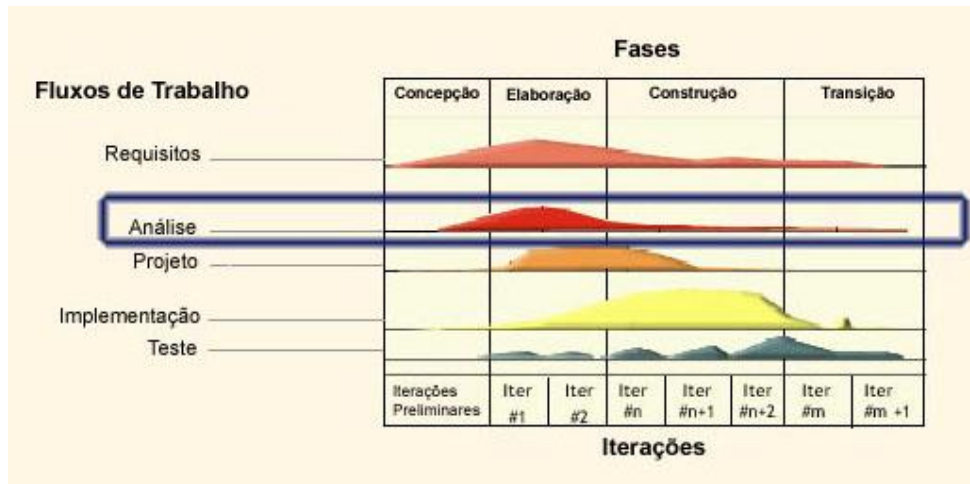


Figura 3.5 – Fluxo de Análise.

O modelo de análise cresce incrementalmente a medida em que casos de uso são analisados. Para cada iteração, é selecionado um conjunto de casos de uso que são realizados no modelo de análise. O sistema é construído como uma estrutura de classes de análise e relacionamentos entre estas classes. Assim, na próxima iteração outro conjunto de casos de uso é realizado e adicionado ao produto da iteração prévia.

Uma maneira prática para realizar este fluxo de trabalho é primeiro identificar e detalhar os casos de uso para uma iteração, e depois, através da análise da descrição de cada caso de uso, sugerir quais classes e relacionamentos são necessários para realiza-lo. Isto é feito para todos os casos de uso em uma iteração.

Dependendo da etapa em que o ciclo de vida do processo se encontra e do tipo de iteração que está sendo trabalhada, a arquitetura do sistema poderá auxiliar a identificação de novas classes e a reutilização de outras já existentes. Cada classe de análise possui um ou vários papéis em uma descrição de um caso de uso. Cada papel especifica as responsabilidades, atributos, e tudo mais que for necessário para detalhar um caso de uso.

Ao estruturar os requisitos do sistema, o modelo de análise fornece uma estrutura com foco na manutenção dos mesmos. Porém, esta estrutura não é útil apenas à manutenção de requisitos, mas também serve de entrada para os

fluxos de projeto e implementação. Desta forma, o modelo de análise pode ser visto como o primeiro passo para o desenvolvimento do modelo de projeto, muito embora, seja considerado um modelo em particular. O fato de se preservar a estrutura do modelo de análise no projeto, permite que o processo desenvolva um sistema de fácil manutenção, que possua um grande poder de recuperação mediante a mudança de requisitos e que inclua elementos que possam ser reutilizados em sistemas relacionados em desenvolvimento.

De qualquer modo, é importante notar que o modelo de análise não trata problemas que são solucionados em modelos de projeto ou implementação. Por causa disso, a estrutura fornecida pelo modelo de análise nem sempre deve ser preservada, mas sim transformada durante o projeto e implementação do sistema. A razão pela qual a preservação da estrutura de um modelo de análise não é efetuada na prática é evidenciada pelo fato do projeto considerar a plataforma de implementação do sistema como, linguagens de programação, sistemas operacionais, *frameworks* pré-fabricados, etc. Portanto, visando a diminuição de custos e o melhor aproveitamento de prazos, a arquitetura deve ser alcançada modificando-se o modelo de análise durante a transição para o modelo de projeto.

3.2.2.3 – Projeto

No fluxo de projeto o sistema é moldado e sua forma é definida de maneira a suprir as necessidades especificadas pelos requisitos. Este fluxo de trabalho define um modelo de projeto que é construído com base no modelo de análise definido no fluxo anterior.

Porém, ao contrário do fluxo de análise, cujo produto é um modelo que descreve características comportamentais e estruturais do sistema em um nível conceitual, o fluxo de projeto desenvolve o modelo de projeto que descreve o sistema em um nível físico, tendo como principal função obter uma compreensão detalhada dos requisitos do sistema, levando em consideração fatores como linguagens de programação, sistemas operacionais, tecnologias de banco de dados, interface com o usuário, etc.

O fluxo de projeto possui seu enfoque entre o fim da fase de elaboração e o início da fase de construção, como pode ser visto na figura 3.6. Isso contribui para a definição de uma arquitetura estável e para a criação do modelo de projeto que servirá de base para o fluxo de implementação. Desta forma, sendo o modelo de projeto bem ligado ao fluxo de implementação, é normal que o modelo de projeto seja mantido através do ciclo de vida completo do sistema.

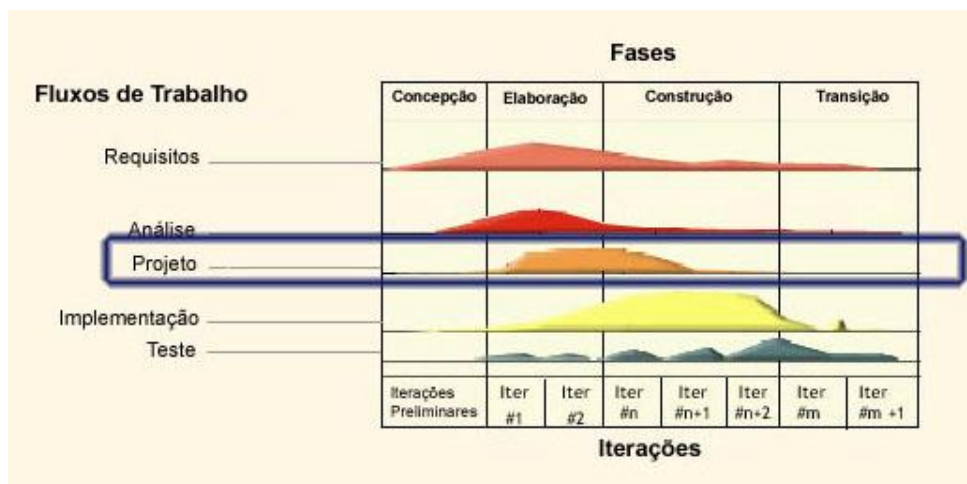


Figura 3.6 – Fluxo de Projeto.

Através do modelo de projeto, casos de uso são realizados por artefatos de projeto representados por ferramentas de modelagem também utilizadas no fluxo de análise como diagramas de classes, diagramas de interação e diagramas de gráfico de estados, agora com o intuito de capturar os requisitos de implementação; ou seja, estes mesmos diagramas são construídos em um nível mais físico que conceitual. Além disso, o fluxo de projeto também utiliza outras ferramentas, não comuns ao fluxo de análise, como diagrama de objetos.

Assim, o modelo de projeto descreve as realizações físicas de casos de uso considerando como requisitos, e outros detalhes relacionados ao ambiente de implementação, causam impacto ao sistema sob consideração. Desta forma, o modelo de projeto funciona como uma abstração da implementação do sistema. Em suma, enquanto que o fluxo de análise se interessa por *o que* o

sistema deve fazer, o fluxo de projeto diz respeito a *como* os requisitos serão implementados e, portanto, pressupõe uma infra-estrutura de implementação e é fortemente influenciado por ela.

O projeto pode dividir o sistema em subsistemas visando a organização dos artefatos do modelo de projeto e partes mais gerenciáveis. Um subsistema pode ser composto de classes de projeto, realizações de casos de uso, interfaces, e recursivamente, outros subsistemas. Além disso, um subsistema pode fornecer interfaces que representam a funcionalidade que elas exportam em termos de operações.

Um subsistema deve ser coesivo, ou seja, seu conteúdo deve estar extremamente relacionado. Por outro lado cada subsistema deve depender o mínimo possível de outro subsistema. A grande vantagem de se trabalhar com subsistemas no fluxo de projeto é que estes subsistemas poderão ser projetados separadamente e concorrentemente por diferentes equipes de desenvolvimento em diferentes níveis do projeto.

3.2.2.4 – Implementação

O fluxo de implementação é baseado no produto do fluxo de projeto, o modelo de projeto; e implementa o sistema em termos de componentes, ou seja, código fonte, arquivos executáveis, etc.

A maior parte da arquitetura do sistema é definida durante o projeto, como pode ser visto na figura 3.7. Portanto, o fluxo de implementação produz um modelo de implementação que se limita a:

- Planejar as integrações do sistema em cada iteração. Neste caso, o resultado é um sistema que é implementado como uma sucessão de etapas pequenas e gerenciáveis.
- Implementar os subsistemas encontrados durante o projeto.
- Testar as implementações e integrá-las, compilando-as em um ou mais arquivos executáveis, antes de enviá-los ao fluxo de teste.

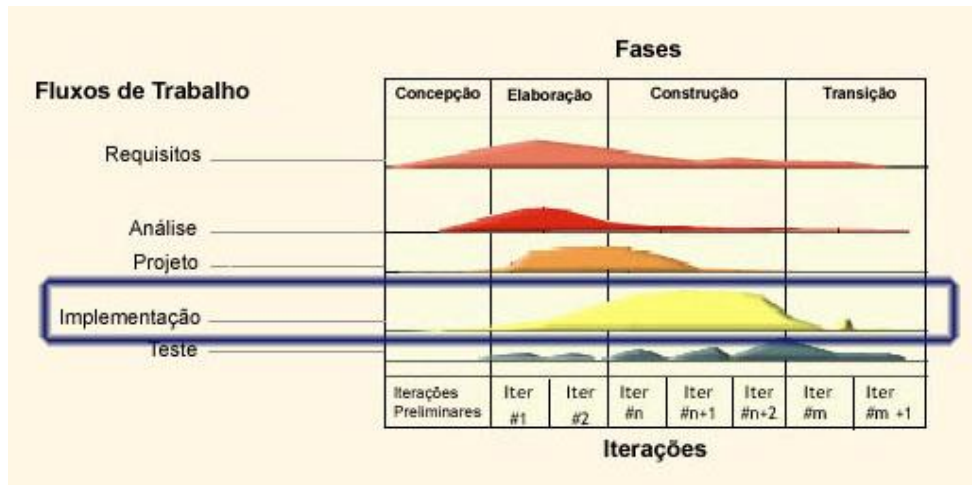


Figura 3.7 – Fluxo de Implementação.

O fluxo de implementação tem maior importância na fase de construção e apesar de ter suas características próprias, a maior parte de suas atividades é realizada de forma quase mecânica, pelo fato das decisões mais difíceis terem sido tomadas durante o fluxo de projeto. Desta forma, o código gerado durante a implementação, deve ser uma simples tradução das decisões de projeto em uma linguagem específica.

3.2.2.5 – Teste

O fluxo de teste é desenvolvido com base no produto do fluxo de implementação, ou seja, os componentes executáveis são testados exaustivamente no fluxo de teste para então serem disponibilizados aos usuários finais. Desta forma o principal propósito do fluxo de teste é realizar vários testes e sistematicamente analisar os resultados de cada teste. Componentes que possuírem defeitos retornarão a fluxos anteriores como os fluxos de projeto e implementação, onde os problemas encontrados poderão ser corrigidos.

O teste de um sistema, propriamente dito, é primeiramente empregado durante a fase de elaboração, quando a arquitetura do sistema é definida, e durante a fase de construção quando o sistema é implementado. Porém, um planejamento inicial de testes pode ser feito durante a fase de concepção. Na

fase de transição, o fluxo de testes se limita ao conserto de defeitos encontrados durante a utilização inicial do sistema. O papel do fluxo de teste dentro do ciclo de vida do processo unificado pode ser visualizado na figura 3.8.

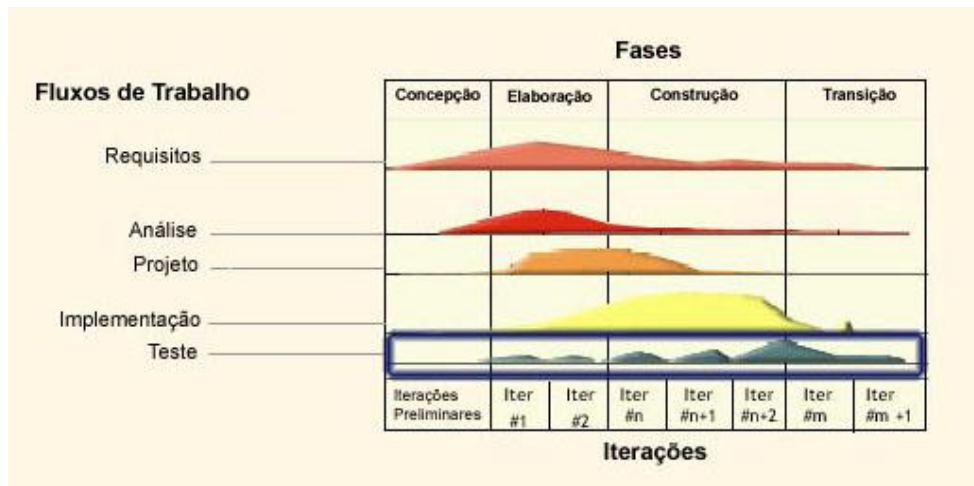


Figura 3.8 – Fluxo de Teste.

O produto do fluxo de teste é o modelo de teste, esse modelo primeiramente descreve como componentes executáveis, provenientes do fluxo de implementação, são testados. O modelo de testes também pode descrever como aspectos específicos do sistema são testados, como por exemplo, se a interface com o usuário é útil e consistente ou se o manual do usuário cumpre seu objetivo.

Em suma, o papel do fluxo de teste é verificar se os resultados do fluxo de implementação cumprem os requisitos estipulados por clientes e usuários, para que possa ser decidido se o sistema necessita de revisões ou se o processo de desenvolvimento pode continuar.

3.2.3 – Fases

Um ciclo está dividido em fases, cada qual podendo ser subdividida em iterações e conseqüentes incrementos. O final de uma fase é marcado por um ponto de verificação, isto é, pela disponibilidade de um conjunto de artefatos

que possibilitem a avaliação do projeto, tais como modelos e outros documentos.

Os pontos de verificação servem a diversos propósitos:

- Gerentes devem tomar certas decisões cruciais antes do trabalho continuar.
- Permitem a monitoração do progresso dos trabalhos.
- Observando o tempo e o esforço despendidos em cada fase, é possível reunir dados úteis para estimar os requisitos de tempo de outros projetos.

Conforme mostra a figura 3.2, cada fase é subdividida em iterações e cada iteração executa as cinco atividades listadas na coluna à esquerda (requisitos, análise, projeto, implementação e testes). As curvas não devem ser interpretadas literalmente, mas representam uma aproximação do esforço despendido com cada atividade em cada fase.

São quatro as fases que compõem o ciclo de vida do Processo Unificado:

3.2.3.1 - Concepção

O objetivo principal da fase de concepção é delimitar o escopo do projeto, definindo como o sistema será utilizado por cada usuário, através da criação dos casos de uso mais relevantes. A partir deste escopo, poderá ser definido o que o desenvolvimento do projeto deverá cobrir, no que diz respeito a custos, prazos, retorno de investimento, etc.

Além disso, o esforço empenhado na fase de concepção poderá evitar o fracasso do projeto através da identificação prévia de riscos. A causa do fracasso de vários projetos é o fato de riscos críticos serem encontrados tarde demais, geralmente durante a integração do sistema ou durante a etapa de testes do mesmo, quando não se há mais tempo nem orçamento para revisões.

A maior parte do trabalho da fase de concepção está concentrado no fluxo de requisitos, porém cada fluxo de trabalho possui seu papel dentro desta fase.

No final da fase de concepção, os objetivos do ciclo de vida do projeto devem ser analisados para se decidir se o desenvolvimento deve prosseguir em plena escala.

3.2.3.1.1 - Requisitos:

O fluxo de requisitos da fase de concepção é caracterizado pela identificação dos requisitos mais relevantes para a definição do escopo do sistema e para a compreensão preliminar da arquitetura do mesmo.

Ao capturar os requisitos nesta fase, o analista de sistemas identifica os casos de uso e atores que definem o escopo do sistema, priorizando e detalhando aqueles que serão, eventualmente, mais importantes para a descrição inicial da arquitetura do mesmo. O resultado deste trabalho é o primeiro modelo de casos de uso

3.2.3.1.2 - Análise:

No fluxo de análise da fase de concepção, o modelo de análise inicial é criado a partir do detalhamento, em termos de classes e relacionamentos, de boa parte dos casos de uso definidos durante o fluxo de requisitos anterior. Este modelo será importante para se obter uma compreensão clara dos casos de uso e para o desenvolvimento da arquitetura do sistema.

Porém, durante a fase de concepção muito pouco do modelo completo de análise é construído, assim o modelo de análise nesta fase é apenas o primeiro passo para a obtenção da visão arquitetural do sistema.

3.2.3.1.3 - Projeto:

A meta do fluxo de projeto nesta fase é fazer um esboço do modelo de projeto que contribuirá para a descrição inicial da arquitetura do sistema.

Isto é feito através da identificação de classes de projeto e seus relacionamentos a partir do estudo dos casos de uso detalhados no fluxo de trabalho anterior, o fluxo de análise.

3.2.3.1.4 - Implementação e Teste:

Para reduzir ao mínimo custos e tempo despendidos na fase de concepção as tarefas relacionadas aos fluxos de implementação e teste podem não ser executadas.

Esta decisão não acarreta nenhum prejuízo ao processo de desenvolvimento do sistema, tendo em vista que as principais metas da fase de concepção, que são a definição do escopo do sistema e a descrição inicial de sua arquitetura, são alcançadas entre os fluxos de requisitos e projeto realizados anteriormente.

3.2.3.2 - Elaboração

Na fase de elaboração os requisitos remanescentes (a maioria) são capturados e transformados em casos de uso; a base da arquitetura, que irá guiar os trabalhos nas fases de construção e transição, é estabelecida; e detalhes adicionais do projeto são averiguados.

Para alcançar seu objetivo, o sistema deve ser estudado mais amplamente do que profundamente, ou seja, nesta fase deve ser concebida uma visão abrangente do sistema, sem a necessidade de detalhes minuciosos.

O foco da fase de elaboração está na formulação de uma base para a arquitetura do sistema. Isso envolve o estudo da maior parte dos casos de uso envolvidos, cerca de 80%.

No final desta fase, estarão definidos o escopo e os objetivos detalhados do sistema, a escolha da arquitetura e a solução para os principais riscos. Assim, as informações necessárias para a fase de construção estarão disponíveis.

3.2.3.2.1 - Requisitos:

A primeira tarefa do fluxo de requisitos da fase de elaboração é identificar os casos de uso adicionais, ou seja, aqueles que não foram identificados na fase de concepção. Estes casos de uso juntos formam cerca de 80% do total de casos de uso referentes ao sistema, porém, só uma parte deste conjunto deve ser detalhada nesta fase, mais especificamente, a parte que corresponde aos casos de uso que contribuem para a compreensão plena dos requisitos e para a criação da base da arquitetura do sistema.

Assim como na fase anterior, os casos de uso capturados neste fluxo deverão ser detalhados e priorizados, porém, a quantidade dos casos de uso identificados e detalhados neste primeiro momento depende das circunstâncias nas quais o sistema estará sendo desenvolvido. Por exemplo, se for estipulado um preço fixo ao cliente, conseqüentemente, a maior parte dos casos de uso deverá ser detalhada, provavelmente mais que 80 %. Se o empreendimento for financiado pelos próprios desenvolvedores, então, a porcentagem dos casos de uso detalhados poderá ser menor. Neste último caso, o risco a ser considerado no desenvolvimento é maior, em compensação, o tempo gasto e o esforço empregado na fase de elaboração são menores. Esta estratégia é adotada quando se espera um retorno imediato do empreendimento.

A partir da identificação e detalhamento dos casos de uso nesta primeira etapa, o analista de sistemas poderá estruturar o modelo de casos de uso. Isto é feito a partir da identificação de similaridades e eliminação de eventuais redundâncias existentes no modelo de casos de uso, de forma a deixá-lo mais compreensivo e de fácil manutenção.

3.2.3.2.2 - Análise:

O trabalho do fluxo de análise da fase de elaboração se concentra na análise dos casos de uso significantes à arquitetura do sistema. Estes casos de uso, que representam um pouco menos de 10% do total de casos de uso definidos, são analisados visando à complementação do trabalho de análise arquitetural feito na fase de concepção. Assim, tendo como ponto de partida a

arquitetura definida superficialmente durante a fase de concepção, este fluxo de análise poderá definir uma base sólida para a arquitetura do sistema de forma a se conseguir uma arquitetura executável.

Através deste trabalho de análise, os casos de uso significantes à arquitetura do sistema são analisados em termos de classes e a estas classes são alocadas responsabilidades específicas. Os relacionamentos existentes entre classes e os atributos de cada classe também são definidos durante esta etapa do processo. Desta forma, as classes que são relevantes à arquitetura são selecionadas e reunidas através da análise de suas responsabilidades, formando pacotes que contribuirão para a definição de uma base para a visão arquitetural do modelo de análise.

Outros casos de uso que não estão diretamente ligados à arquitetura do sistema e que não são relevantes para a compreensão preliminar dos requisitos são analisados posteriormente na fase de construção.

3.2.3.2.3 - Projeto:

Nesta fase menos de 10% do total de casos de uso são projetados e implementados. Esta pequena porcentagem é apenas uma fração dos casos de uso identificados nesta fase. O projeto da fase de elaboração é realizado a nível arquitetural, ou seja, o projeto envolve classes, subsistemas e casos de uso significantes à arquitetura.

Tanto pacotes na análise quanto subsistemas no projeto são importantes para a definição de uma visão arquitetural. Apesar de várias classes poderem não ser significantes à arquitetura, pacotes e subsistemas geralmente são.

O arquiteto é o responsável pelo projeto dos aspectos significantes à arquitetura, ele continua o trabalho iniciado na fase de concepção e projeta a arquitetura em camadas.

As camadas mais baixas da arquitetura são caracterizadas por representarem mecanismos de projeto, ou seja, mecanismos do sistema operacional no qual o sistema proposto irá operar, linguagens de programação, sistemas de banco de banco de dados, etc.

As camadas mais altas estão próximas às camadas de aplicação da arquitetura. Desta forma, baseado nos pacotes definidos no modelo de análise, o arquiteto identifica os subsistemas correspondentes que serão incluídos no modelo de projeto. Geralmente, cada pacote do modelo de análise se torna um subsistema no modelo de projeto, porém nem sempre isto acontece, algumas vezes um pacote pode se referir a vários subsistemas do modelo de projeto, e em outros casos, um subsistema pode ser criado com base em vários pacotes do modelo de análise.

Assim como na fase anterior, classes de projeto são identificadas através da “tradução” de classes de análise, porém, durante a fase de elaboração, só são selecionadas as classes de projeto interessantes à arquitetura, visando o enriquecimento da descrição arquitetural do projeto.

Durante o fluxo de projeto, a realização de cada caso de uso é feita em um nível mais físico do que conceitual, ou seja, classes de projeto, subsistemas, interfaces, a linguagem de programação e o banco de dados são levados em consideração neste momento. O resultado desta atividade é um conjunto de realizações de casos de uso de projeto, onde são criados artefatos para cada caso de uso significativo à arquitetura.

3.2.3.2.4 - Implementação:

O fluxo de trabalho de implementação na fase de elaboração implementa e testa os elementos significantes à arquitetura com base no modelo de projeto resultante do fluxo anterior.

Desta forma, são identificados os componentes necessários para a implementação dos subsistemas predefinidos. Este trabalho prossegue de maneira incremental até que se obtenha a primeira versão executável do sistema proposto.

3.2.3.2.5 - Teste:

Durante este fluxo de trabalho são testados os subsistemas executáveis produzidos no fluxo de implementação.

O teste se inicia pela camada mais baixa da arquitetura, ou seja, o banco de dados. No caso das camadas mais altas, o mais importante é avaliar como estas utilizam as mais baixas.

Os testes realizados não se limitam a avaliar apenas a funcionalidade do sistema, mas também sua performance.

3.2.3.3 - Construção

O trabalho da fase de construção se inicia com base na arquitetura executável, produzida durante a fase de elaboração, e prossegue através de iterações e incrementos, com objetivo de desenvolver um produto pronto para operações iniciais no ambiente de usuário, ou seja, a versão beta.

Mais especificamente, durante a fase de construção são detalhados os casos de uso remanescentes e a descrição arquitetural é modificada quando necessário. Os fluxos de trabalho prosseguem através de iterações adicionais, preenchendo os modelos de análise, projeto e implementação. Subsistemas são integrados e testados, da mesma forma que o sistema como um todo.

Enquanto as fases de concepção e elaboração estão ligadas diretamente à modelagem do sistema, a fase de construção é caracterizada pelo desenvolvimento, isto é, a construção de um sistema ou produto dentro dos parâmetros de custo e prazos.

3.2.3.3.1 - Requisitos:

Durante o fluxo de requisitos da fase de elaboração, foram identificados todos os casos de uso e atores, porém, apenas uma parcela destes casos de uso, aqueles necessários para a descrição da base da arquitetura, foram detalhados. Agora na fase de construção, deverão ser detalhados os casos de uso remanescentes.

Outra tarefa do fluxo de requisitos da fase de construção consiste na construção de protótipos de interfaces com usuários, isto é necessário quando a interface em questão for muito complexa.

De qualquer forma o trabalho do fluxo de requisitos exercido nesta fase é pequeno em comparação ao trabalho de implementação, que é onde esta fase foca suas atividades.

3.2.3.3.2 - Análise:

Na prática, o modelo de análise dificilmente será preservado até esta etapa do processo, transferindo ao fluxo de projeto a responsabilidade da análise dos casos de uso remanescentes, de forma a proporcionar mais objetividade ao processo de desenvolvimento. Porém, esta decisão fica a critério do analista de sistemas.

Se o modelo de análise for preservado até esta fase do processo, mesmo será completado através da análise de classes e casos de uso com base no produto do fluxo anterior, ou seja, aquelas classes e casos de uso remanescentes, que não foram analisados na fase de elaboração.

O resultado do fluxo de análise nesta fase é um modelo de análise mais completo que aquele desenvolvido durante a fase de elaboração, fazendo com que a visão arquitetural definida anteriormente seja apenas um subconjunto do modelo de análise atual.

3.2.3.3.3 - Projeto:

Este fluxo é responsável pelo projeto de aproximadamente 90% dos casos de uso, aqueles que não foram utilizados para desenvolver a base da arquitetura durante a fase anterior.

Porém, o projeto de subsistemas só é necessário se for verificado que os subsistemas a serem adicionados são similares ou alternativos aqueles já existentes.

3.2.3.3.4 - Implementação:

Este é o principal fluxo de trabalho da fase de construção, onde o maior esforço da fase é empregado.

O trabalho de implementação é feito com base no modelo de projeto, revisto e atualizado no fluxo anterior, onde a arquitetura do sistema deve estar estabelecida, porém algumas atualizações podem ser necessárias, o que será de responsabilidade do arquiteto. A partir da arquitetura preestabelecida são implementados e testados os subsistemas.

O resultado deste trabalho, depois de algumas iterações, junto com a integração dos subsistemas e teste, é a versão operacional inicial, representado 100% do total de casos de uso.

3.2.3.3.5 - Teste:

O objetivo deste fluxo é a realização de testes na primeira versão operacional do sistema, proveniente do fluxo de implementação anterior.

Os testes realizados devem seguir procedimentos preestabelecidos com o intuito de alcançar metas estipuladas em um plano de testes.

Se um teste não alcançar seus objetivos os procedimentos devem ser modificados até que os testes sejam realizados de forma satisfatória.

3.2.3.4 - Transição:

A fase de transição possui como objetivo estabelecer o produto no ambiente operacional. A forma com a qual o projeto cria seu foco varia com a natureza da relação do produto com o mercado. Por exemplo, se um produto for lançado no mercado, a equipe do projeto distribui uma versão beta para que seja avaliado por clientes em geral. Se um produto for desenvolvido para um único cliente, a equipe disponibiliza apenas uma cópia da versão beta.

A partir da avaliação da versão beta do sistema, a equipe de desenvolvimento pode verificar se o sistema realmente cumpre as necessidades do usuário, se possui falhas, problemas e se há ambigüidades na documentação do usuário. Além disso, pode ser verificado se os usuários encontram dificuldades na utilização do sistema e se, desta forma, necessitam de treinamento.

Dependendo dos resultados, a equipe de desenvolvimento pode modificar o sistema ou seus respectivos artefatos.

Nesta fase, o objetivo não está na reformulação do produto, pelo fato das mudanças significativas exigidas pelo cliente, já terem sido incorporadas nos fluxos de requisitos das fases anteriores. Portanto, a fase de transição procura por deficiências mínimas que passaram despercebidas pela fase de construção e possam ser corrigidas dentro da arquitetura existente.

Outra tarefa da fase de transição é a disponibilização de treinamentos que possibilitaram que o cliente utilize o produto de forma eficiente.

A conversão de bases de dados antigas para a nova configuração é também de responsabilidade da fase de transição.

A fase de transição se conclui com a entrega do produto.

3.2.3.4.1 - Atividades da fase de transição

São poucas as atividades exercidas pelos cinco fluxos de trabalho nesta fase, pelo fato da maior parte do trabalho já ter sido realizado durante a fase de construção. Assim, o foco dos fluxos de trabalho está na correção de defeitos visando à eliminação de falhas que possam ocorrer na utilização inicial do produto, e na realização de testes para assegurar que estas correções não provocaram novos defeitos.

Porém, o maior esforço da fase de transição está na realização de atividades que não estão diretamente relacionadas aos cinco fluxos de trabalho do processo unificado.

Em geral, as atividades da fase de transição são:

- Preparar a versão beta do produto,
- Instalar a versão beta para que seja testada pelos usuários,
- Gerenciar o resultado dos testes,
- Adaptar o produto corrigido às circunstâncias definidas pelo usuário,
- Completar os artefatos do projeto,
- Determinar quando o projeto chegará a sua conclusão.

Esta seqüência de atividades varia de acordo com a natureza de cada projeto, ou seja, se o produto está sendo desenvolvido para o mercado ou para um cliente específico. No primeiro caso, haverá muitos usuários em potencial, onde cada um utilizará o produto de forma específica, sem cumprir uma rotina de testes preestabelecida.

No segundo caso, o cliente provavelmente escolherá um local para a instalação inicial do produto, e seguirá uma rotina sistemática para a realização dos testes.

O esquema de atividades varia também dependendo do fato do sistema ter sido desenvolvido visando à substituição de um já existente ou não. No caso de substituição, provavelmente a tarefa de migração ou conversão de dados, do sistema substituído para o novo sistema, deverá ser realizada.

O principal fator que determinará a conclusão da fase de transição, e por conseguinte, do projeto, é a “satisfação” do cliente, o que também deve ser avaliada sob algumas considerações.

No caso do produto ser lançado no mercado, o gerente do projeto considera que a maior parte de usuários estará satisfeita quando o projeto disponibilizar uma versão do produto que apresente soluções para os problemas encontrados durante os testes da versão beta. Na maioria dos casos, o produto continua evoluindo. Sendo assim, a fase de transição terminará quando o projeto passar a responsabilidade de manutenção contínua para a equipe de suporte.

No caso do produto ser desenvolvido para um cliente em particular, o gerente de projeto considera que o cliente estará satisfeito quando o sistema concluir seus testes de forma satisfatória. Isto depende da interpretação dos requisitos estipulados no contrato original e nas eventuais mudanças feitas nos requisitos durante fases posteriores. O cliente pode contratar o serviço de suporte do vendedor do sistema, assumir a responsabilidade pelo suporte ou delega-la a terceiros.

Os detalhes que evidenciam a satisfação do cliente podem variar, porém, uma vez que o projeto alcança seus objetivos de forma satisfatória, a fase de transição estará concluída. Caso contrário, deverá se iniciar um novo ciclo de desenvolvimento.

Capítulo 4 – Estudo de Casos

O estudo de casos proposto neste trabalho diz respeito ao Sistema Informatizado de Gestão do Laboratório Central da Eletronorte, o SIGLacen.

Este capítulo mostrará os fluxos de trabalho que contribuíram para a modelagem do sistema através das fases do Processo Unificado.

Um estudo de casos a respeito do fluxo de teste do Processo Unificado não é apresentado neste trabalho, tendo em vista que esta etapa é realizada durante o desenvolvimento efetivo do sistema e a partir da disponibilização de um protótipo executável do mesmo. Desta forma, o estudo de casos apresentado neste capítulo, considera que apenas um ciclo de desenvolvimento, composto de fluxos de requisitos, análise, projeto e implementação, é necessário para que o sistema seja concluído, atendendo às necessidades de clientes e usuários.

Além do estudo de casos propriamente dito, este capítulo faz uma abordagem a respeito da organização do Laboratório Central com o propósito de esclarecer alguns termos utilizados.

4.1 – O Laboratório Central da Eletronorte, LACEN.

Criado em 1983, o Lacen é um Departamento da Diretoria de Produção e Comercialização da ELETRONORTE, responsável pela realização de ensaios de alta tecnologia, que por motivos técnico-econômicos devem ser centralizados, pelo gerenciamento do programa de manutenção e calibração dos instrumentos de testes e pela implantação de novas técnicas de ensaios, visando principalmente o estabelecimento de manutenção preventiva. Presta serviços à ELETRONORTE nos diversos estados da Região Amazônica e também à outras empresas do País.

O principal objetivo do Lacen é realizar ensaios, consultorias e treinamentos de natureza elétrica, eletrônica, mecânica e físico-química. Estas atividades são agrupadas em processos respeitando as respectivas áreas de conhecimento e conduzidas por um Líder de Processo, designado para tal.

O corpo de funcionários do Lacen é constituído por gerentes e colaboradores, estes últimos, exercem atividades de sua competência em um ou mais processos. Dentre os gerentes do laboratório encontram-se o gerente da qualidade (GQL), gerente técnico (GTL) e o gerente do Lacen (GL).

4.2 – O Sistema Informatizado de Gestão do Laboratório Central, SIGLacen – Descrição do Sistema.

O Sistema Informatizado de Gestão do Laboratório Central da Eletronorte (SIGLacen) tem como objetivo controlar de forma automatizada as atividades exercidas pelos colaboradores do laboratório, assim como o gerenciamento de informações de pessoal, planejamento estratégico, instrumentos, custos, orçamentos e despesas, oferecendo desta forma, informações de cunho gerencial através da emissão de relatórios e índices estatísticos, que obedecem as normas do sistema de qualidade e os interesses de gerentes e colaboradores, de maneira a alcançar a satisfação de ambos, visto que fará parte do dia a dia para atender as necessidades dos mesmos.

O SIGLacen foi idealizado devido a necessidade de armazenamento consistente e manipulação de informações, a nível gerencial, das atividades executadas pelos colaboradores do Lacen, objetivando a quantificação efetiva dos lucros e despesas do laboratório.

Estas atividades foram estudadas durante o fluxo de captura de requisitos, tendo como principal fonte de informação o Manual da Qualidade, Manual de Procedimentos, colaboradores e gerentes.

O fluxo de análise do sistema foi caracterizado pelo estudo detalhado do modelo de casos de uso, produto do fluxo de captura de requisitos. A análise do sistema permitiu que fosse criado um modelo de análise que serviu de base para o fluxo de projeto onde o sistema foi dividido em cinco subsistemas. São eles:

- Sistema de Gestão de Pessoal (SIGP)
- Sistema de Gestão de Serviços e Instrumentos (SGSI)
- Sistema de Planejamento Estratégico (SIPE)
- Sistema de Controle do Sistema da Qualidade (SCSQ)
- Sistema de Custos, Orçamentos e Despesas (SCOD)

Cada um destes cinco subsistemas aborda uma área específica de atuação do SIGLacen, agrupando atividades de mesma natureza.

Portanto, partindo do princípio de que estes cinco subsistemas se relacionam diretamente e armazenam informações em um único banco de dados, o sistema garante o fornecimento de informações concisas, sem redundâncias, que serão utilizadas tanto a nível operacional quanto tático e estratégico.

Durante o fluxo de implementação o diagrama físico de classes extraído do modelo de projeto foi implementado utilizando-se o sistema gerenciador de banco de dados SQL Server 7.0, possibilitando mais segurança e confidencialidade nas informações das atividades desenvolvidas tanto para clientes internos quanto externos à ELETRONORTE. O modelo físico completo do banco de dados possui 84 tabelas que se relacionam visando à integridade dos dados. A interface com o usuário e as rotinas, procedimentos e funções do sistema foram desenvolvidas através do Visual Basic 6.0 com o auxílio da ferramenta Crystal Report para a elaboração de relatórios que obedecessem as normas do sistema de qualidade. A figura 4.1 mostra a tela principal do SIGLacen em tempo de execução.



Figura 4.1 – Tela Principal do SIGLacen.

Com o sistema implantado, atividades, que hoje são executadas de forma manual, poderão ser automatizadas fazendo com que resultados sejam obtidos de forma mais rápida e eficaz, além disso, informações a respeito do resultado de investimentos do laboratório, poderão ser quantificadas com maior eficácia através de análises estatísticas.

Também deve ser considerado que o sistema executará atividades que antes eram de responsabilidade de colaboradores e/ou de outros sistemas. Desta forma, estes colaboradores poderão ser realocados de acordo com o interesse gerencial e os sistemas obsoletos poderão ser extintos.

Em suma, o SIGLacen, depois de sua implantação, fará parte da rotina dos colaboradores do laboratório, agilizando e otimizando tarefas em variados processos e tirando de uso diversos aplicativos que utilizavam banco de dados

diferentes, garantindo desta forma a consistência de dados e credibilidade de informações.

O estudo de casos proposto neste trabalho não abrange o SIGLacen com um todo, uma abordagem deste nível seria inviável tendo em vista a magnitude do sistema e a natureza do trabalho em questão. Além disso, é desnecessária uma abordagem completa do processo de desenvolvimento do SIGLacen para exemplificar o uso da UML dentro do Processo Unificado.

Desta forma, o estudo de casos apresenta uma parte de um dos cinco subsistemas do SIGLacen, mais especificamente, a Solicitação de Serviços, procedimento que compõe o Subsistema de Serviços e Instrumentos. Além disso, o estudo de casos abrange outros procedimentos que estão diretamente relacionados à Solicitação de Serviços e que, por este motivo, não podem ser omitidos.

4.2.1 – O Sistema de Solicitação de Serviços (SS).

A SS é o sistema utilizado atualmente para armazenar e gerenciar informações a respeito dos serviços solicitados ao Laboratório Central. A manipulação dessas informações é de grande interesse gerencial, objetivando o cálculo de dados estatísticos que indicam os lucros e despesas a respeito dos serviços realizados. Sendo assim, o SIGLacen possui a necessidade implícita de gerenciar essas informações, porém, a pura e simples integração do sistema atual de SS ao SIGLacen é inviável, considerando as diferenças encontradas entre a arquitetura existente da SS atual e a arquitetura proposta para a SS integrante do SIGLacen.

Mesmo se houvesse uma tentativa de integração do sistema atual de SS ao SIGLacen, haveria, ainda assim, a necessidade de programação do sistema de SS atual, para suprir determinados requisitos impostos pelo próprio SIGLacen. Desta forma, a remodelagem de um novo sistema de SS, dentro do escopo do SIGLacen, pouparia mais tempo, esforço e investimento do que a tentativa de reutilização do sistema atual.

Assim, o sistema de SS atual foi a principal fonte de requisitos utilizada para a modelagem do novo sistema de solicitação de serviços.

4.2.2 – A Solicitação de Serviços.

Um cliente, interno ou externo à Eletronorte, pode solicitar um serviço ao Laboratório Central. Porém, se o cliente for externo, o serviço deverá ser formalizado por um contrato, caso contrário, o serviço não precisará nem deverá ser intermediado por um contrato, tendo em vista que, neste último caso, o serviço será realizado para a própria Eletronorte.

Depois de efetivada a solicitação, o tratamento da mesma é realizado em duas etapas e possui algumas considerações. A primeira etapa consiste no tratamento de informações do serviço a nível administrativo. Quando um serviço é solicitado ao Lacen um colaborador do processo de administração é encarregado de verificar se o serviço possui um contrato e se este contrato tem como objeto um material a ser analisado, mesmo que o serviço não possua um contrato, ou seja, tenha sido solicitado por um cliente interno, a existência de um material para análise, ou seja, um item de ensaio, deve ser averiguada. Caso exista um item de ensaio, o respectivo colaborador deve seguir o procedimento de recepção, identificação e manuseio de itens de ensaio (PRIE) existente no manual de procedimentos do Lacen. A partir do atendimento das normas estipuladas no manual de procedimentos, o colaborador responsável deve cadastrar as respectivas informações do serviço no sistema de SS, determinando o número do contrato (se existir), o cliente, o material objeto do serviço (se existir), o processo no qual este serviço será realizado inicialmente, o tipo de serviço e a condição inicial do serviço. Logo após salvar estas informações, o sistema criará um número de identificação para o serviço, número este que servirá como referência para a próxima etapa.

A segunda etapa de tratamento de uma solicitação de serviço é realizada a nível operacional. Quando um serviço é encaminhado a um determinado processo, um ou mais colaboradores são responsáveis pela realização do mesmo, esses colaboradores, neste momento, assumem o papel de executores do serviço. Após a realização do serviço, informações referentes a procedimentos técnicos, tempo e custos são registradas pelo programa de

SS, complementando as informações administrativas do serviço inseridas anteriormente.

A partir destas informações armazenadas, o sistema de SS poderá gerar relatórios estatísticos de grande relevância sob o ponto de vista gerencial, e desta forma, contribuir para tomada de decisões estratégicas.

4.3 – O Estudo de Casos propriamente dito

4.3.1 – Fase de Concepção

Durante esta fase do Processo Unificado, devemos delimitar o escopo do projeto, definindo como o sistema será utilizado por cada usuário, através da criação dos casos de uso mais relevantes.

Com base no Modelo de Casos de uso inicial poderemos desenvolver os primeiros modelos de análise e projeto, a partir da criação de diagramas de classes conceituais e lógicos, respectivamente.

Esta fase é apresentada em uma única interação.

4.3.1.1 – Interação #1

4.3.1.1.1 - Fluxo de Requisitos

A partir das considerações feitas no início deste capítulo podemos chegar ao diagrama principal de Casos de Uso apresentado na figura 4.2.

Este diagrama é o primeiro passo no desenvolvimento do Modelo de Casos de Uso. A partir deste diagrama principal, o modelo de casos de uso crescerá incrementalmente, através do refinamento dos casos de uso, até alcançar o objetivo desta fase, delimitar o escopo do sistema sob a perspectiva de seus usuários.

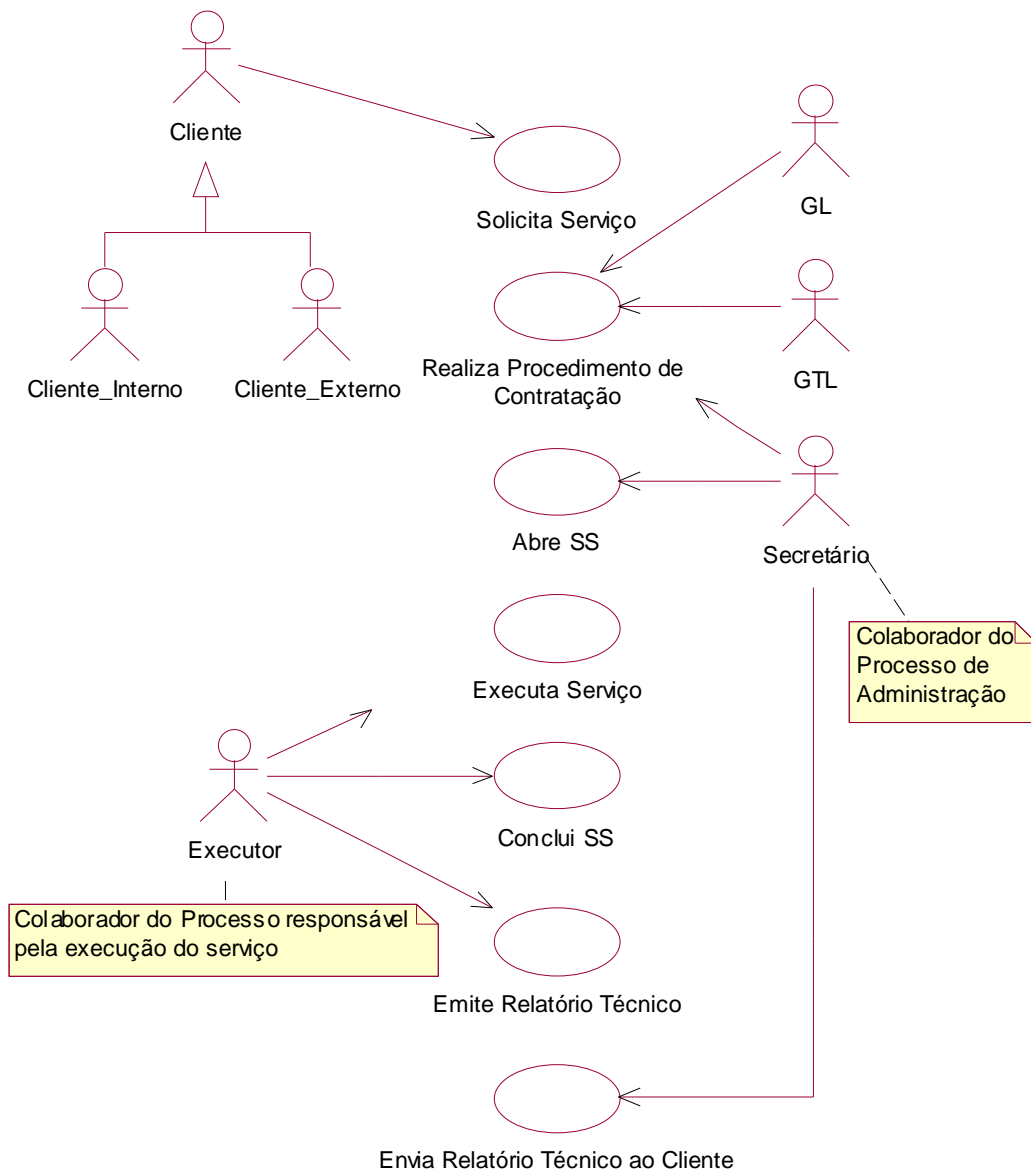


Figura 4.2 – Diagrama Principal de Casos de Uso.

O primeiro refinamento do diagrama principal de Casos e uso é realizado com o objetivo de detalhar o caso de uso 'Realiza Procedimento de Contratação', e é apresentado na figura 4.3.

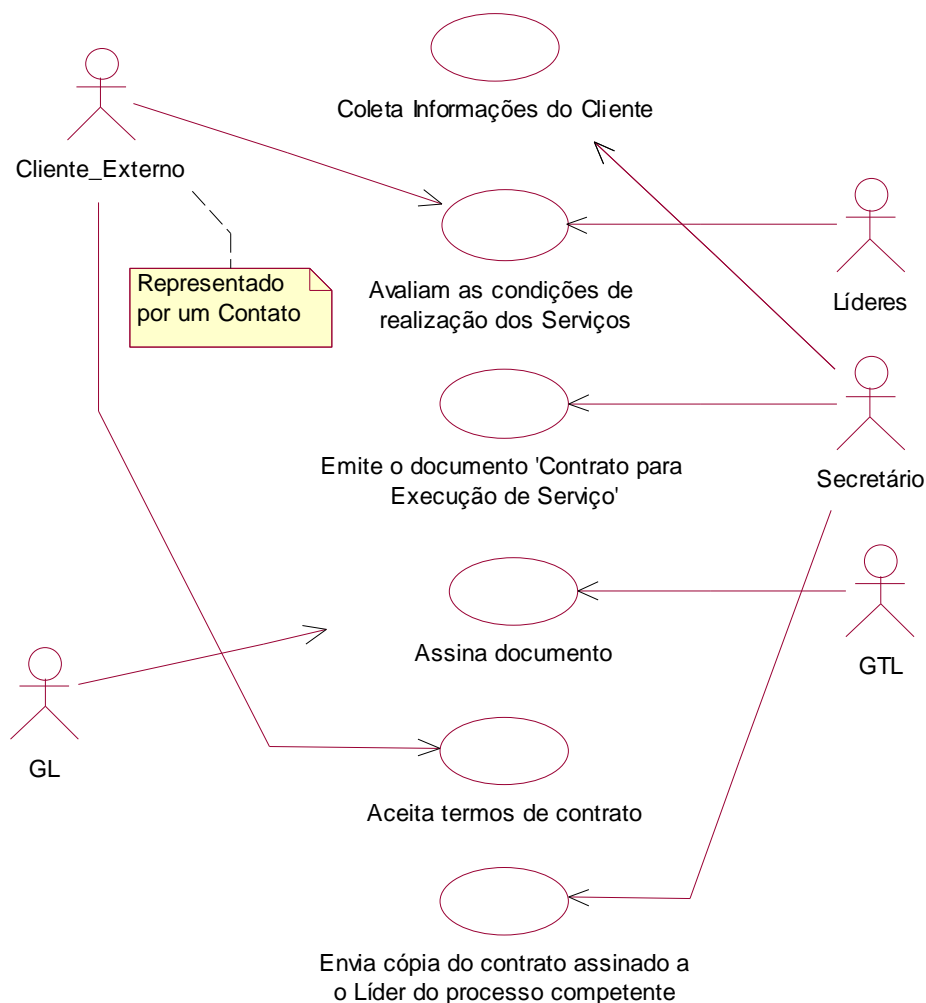


Figura 4.3 – Diagrama de Casos de Uso 'Refinamento do Caso de Uso Realiza Procedimento de Contratação'.

Se neste momento, algum caso de uso apresentado no diagrama da figura 4.3 precisasse ser refinado, visando o objetivo da fase de concepção, a construção de diagramas de casos de uso para o refinamento poderia ser realizada imediatamente ou em outra interação. Porém, nesta fase, não temos a necessidade de refinar o diagrama da figura 4.3, o que nos permite voltar ao diagrama principal da figura 4.2 e detalharmos o caso de uso 'Abre SS', conforme mostra a figura 4.4.

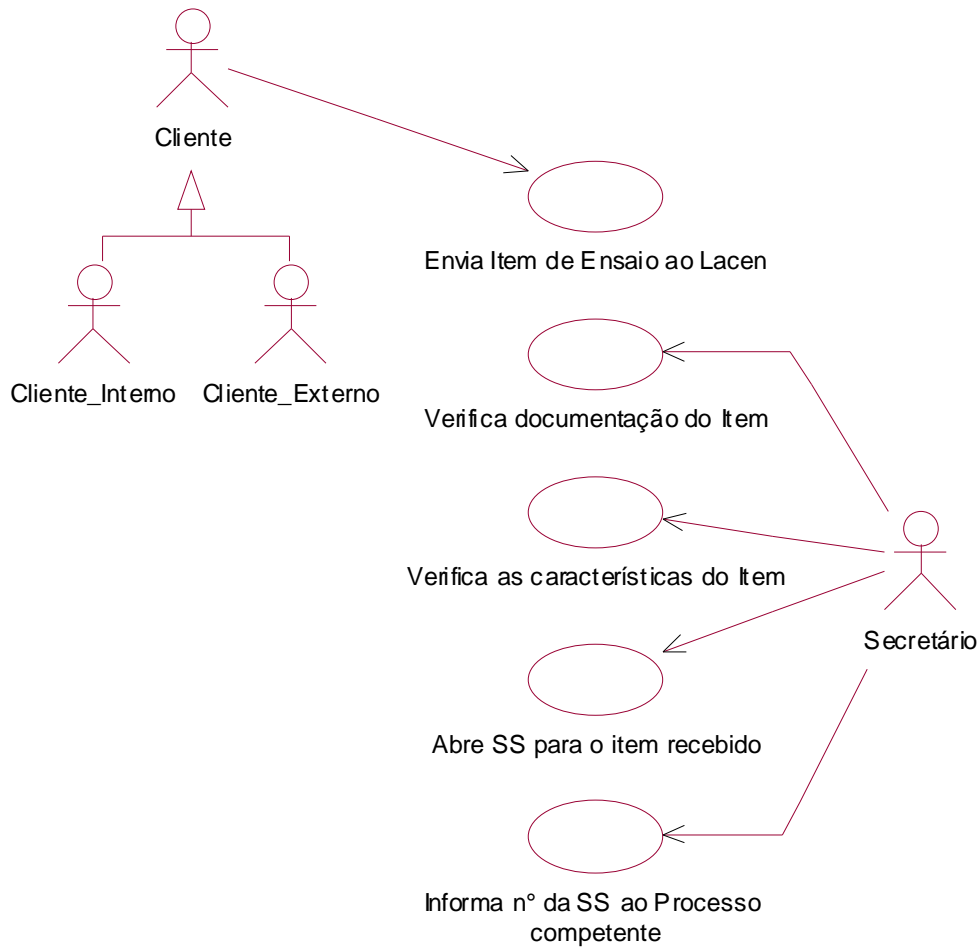


Figura 4.4 – Diagrama de Casos de Uso 'Refinamento do Caso de Uso Abre SS'.

O diagrama da figura 4.4 também não apresenta a necessidade de refinamento de seus casos de uso dentro da fase de concepção. Qualquer trabalho de refinamento deste diagrama seria desnecessário, tendo em vista que detalhes minuciosos na modelagem dos requisitos não são importantes neste momento.

Desta forma, podemos continuar com o refinamento do diagrama de casos de uso principal, através do caso de uso 'Conclui SS', conforme mostra a figura 4.5.

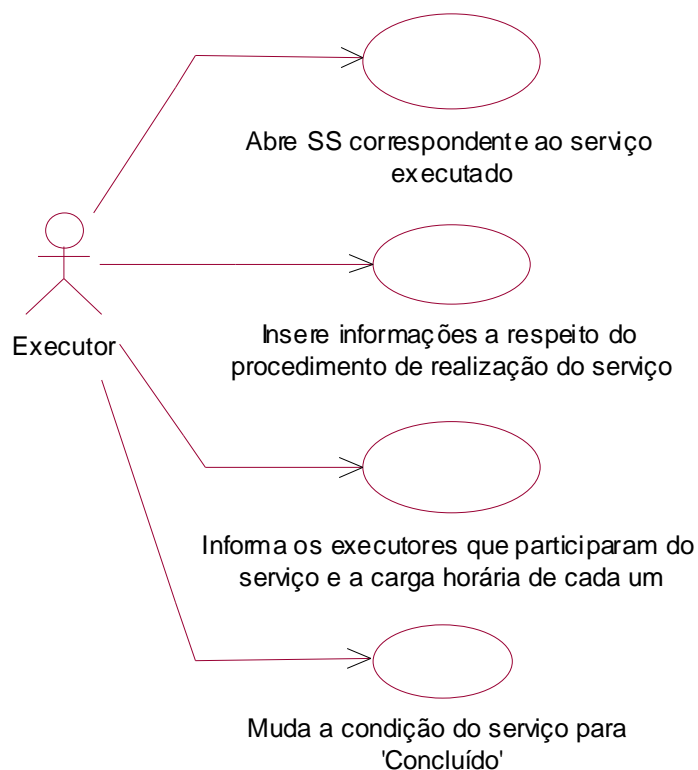


Figura 4.5 – Diagrama de Casos de Uso 'Refinamento do Caso de Uso Conclui SS'.

O fluxo de requisitos da fase de concepção termina com a criação do diagrama da figura 4.5. Desta forma, obtemos o primeiro Modelo de Casos de Uso que servirá de base para a criação dos modelos de análise e projeto dos fluxos seguintes.

4.3.1.1.2 - Fluxo de Análise

O papel do fluxo de análise nesta etapa do processo é identificar classes e seus relacionamentos através do estudo do modelo de casos de uso desenvolvido no fluxo de trabalho anterior. Desta forma, o diagrama de classes desenvolvido neste momento deve ser o mais simples possível, visando apenas à realização em termos de classes e relacionamentos dos casos de uso definidos anteriormente.

O primeiro diagrama de classes do modelo de análise é mostrado na figura 4.6.

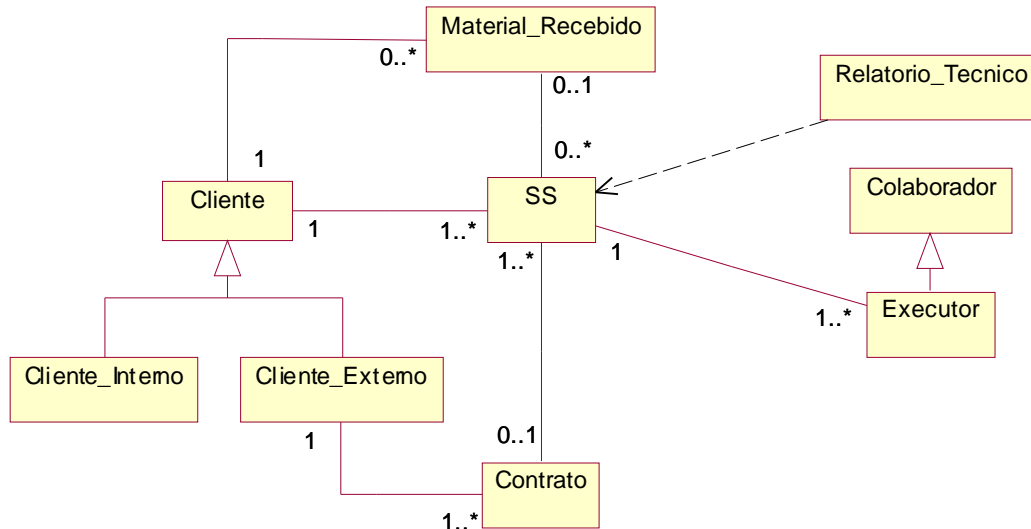


Figura 4.6 – Diagrama de Classes do Modelo de Análise. Fase de Concepção.

Podemos perceber no diagrama da figura 4.6 relacionamentos de associação, generalização e dependência entre classes. Baseado nestes relacionamentos podemos afirmar que, por exemplo, a classe ‘Executor’ é *um tipo de* ‘Colaborador’ e que ‘Relatório Técnico’ depende de ‘SS’. Além disso, as multiplicidades existentes nas extremidades de cada associação nos permitem tirar conclusões de que, por exemplo, cada ‘Cliente_Extimo’ deve possuir no mínimo um ‘Contrato’ e cada ‘Contrato’ diz respeito a somente um ‘Cliente_Extimo’.

O diagrama de classes do fluxo de análise é construído de forma conceitual, não se preocupando, portanto, se a linguagem de programação e o sistema gerenciador de banco de dados escolhidos disponibilizarão recursos para que os conceitos de orientação a objeto possam ser implementados, tais considerações são de responsabilidade do fluxo de projeto.

4.3.1.1.3 - Fluxo de Projeto

O fluxo de projeto da fase de concepção é responsável pela criação de um diagrama de classes lógico, baseado no diagrama de classes conceitual produzido durante o fluxo anterior.

O diagrama de classes do fluxo de projeto é apresentado na figura 4.7.

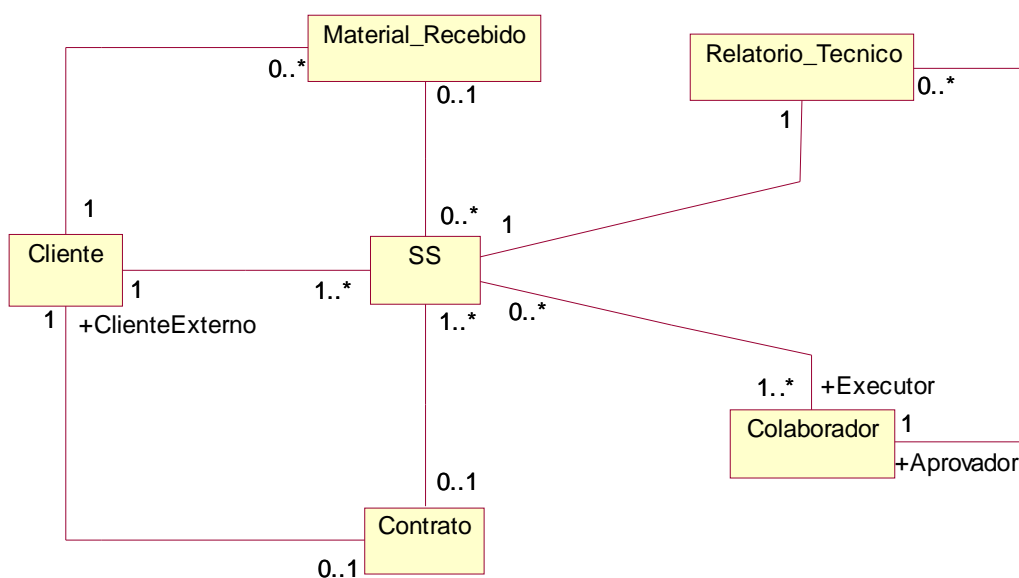


Figura 4.7 – Diagrama de Classes do Modelo de Projeto. Fase de Concepção.

O diagrama de classes da figura 4.7 foi desenvolvido considerando as questões de implementação do sistema. Por este motivo, os relacionamentos que representam conceitos de orientação a objetos no diagrama da figura 4.6 foram substituídos por associações, tendo em vista que o sistema será implementado utilizando-se um banco de dados relacional e uma linguagem de programação visual estruturada, respectivamente SQL Server 7 e Visual Basic 6.

Assim, podemos perceber que, por exemplo, as classes 'Executor', 'Cliente_Interno' e 'Cliente_Externo' unem-se com suas respectivas superclasses, e os relacionamentos que antes possuíam passam a ser caracterizados com papéis, indicando que apenas um subconjunto de cada superclasse obedece o relacionamento.

Desta forma, encerramos a fase de concepção, considerando que o fluxo de implementação desta fase, não se faz necessário neste caso em particular.

4.3.2 – Fase de Elaboração

O principal objetivo da fase de elaboração é estabelecer a base da arquitetura que irá guiar os trabalhos nas fases de construção e transição, isto é feito com base nos requisitos que ainda não foram capturados na fase de concepção.

A fase de elaboração deste estudo de casos é desenvolvida através de duas interações.

4.3.2.1 – Interação #1

4.3.2.1.1 - Fluxo de Requisitos

O fluxo de requisitos da fase de elaboração tem como objetivo a criação de diagramas de casos de uso com base nos requisitos remanescentes, ou seja, aqueles ainda não identificados durante a fase de concepção. A criação destes diagramas incrementará o modelo de casos de uso, o qual servirá de base para o trabalho de complementação dos modelos de análise e projeto nos fluxos seguintes.

O primeiro diagrama de casos de uso criado nesta etapa do processo é apresentado pela figura 4.8.

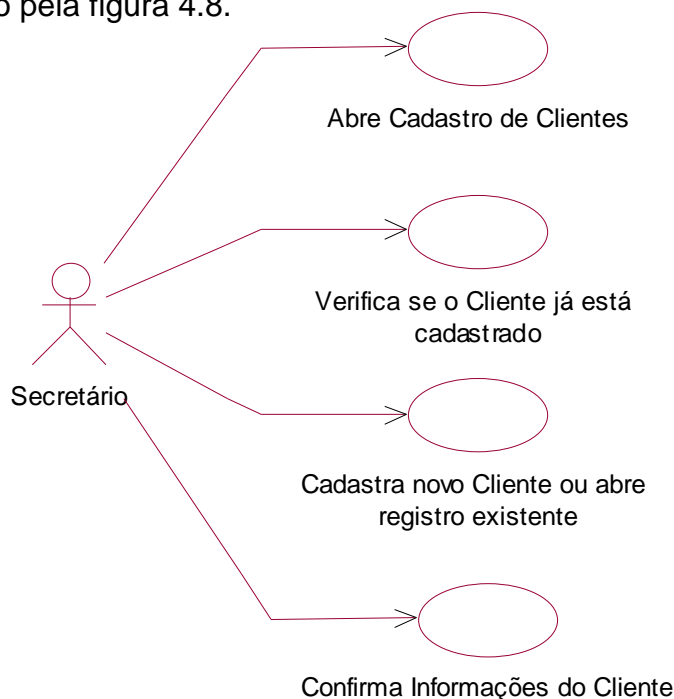


Figura 4.8 – Diagrama de Casos de Uso 'Refinamento do Caso de Uso Coleta informações do Cliente'.

Este diagrama é um refinamento do caso de uso 'Coleta informações do Cliente' existente no diagrama de casos de uso da figura 4.3.

Observando o diagrama da figura 4.8, percebe-se que o mesmo foi desenvolvido em um nível de abstração diferente daquele expresso pelos diagramas de casos de uso da fase de concepção.

Na fase de concepção deste estudo de casos, os diagramas de caso de uso mostram como os atores interagem com os casos de uso que representam procedimentos existentes para a execução de determinadas atividades dentro do Lacen. Na fase de elaboração, os diagramas de casos de uso são desenvolvidos dentro do contexto de utilização do sistema em questão.

Outro diagrama de casos de uso desenvolvido nesta fase é apresentado na figura 4.9.

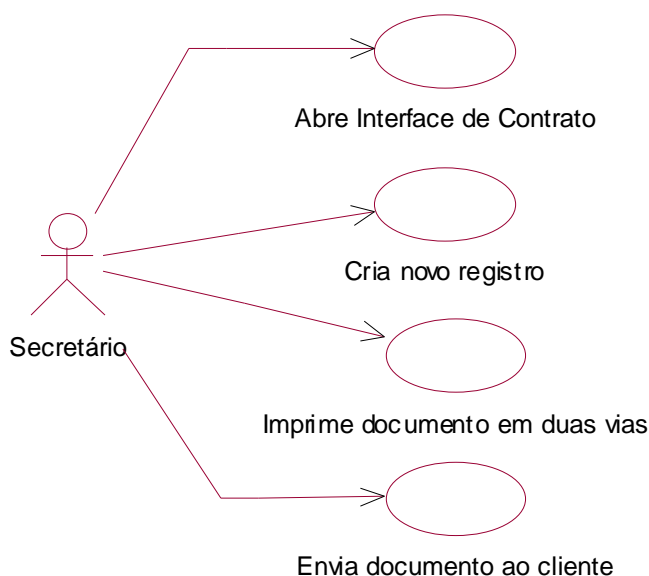


Figura 4.9 – Diagrama de Casos de Uso 'Refinamento do Caso de Uso Emitir o documento Contrato para a Execução do Serviço'.

O diagrama da figura 4.9 é um refinamento do caso de uso 'Emitir o documento Contrato para a execução do Serviço' existente no diagrama de casos de uso da figura 4.3.

Ele expressa o comportamento do usuário, representado pelo ator 'Secretário', dentro do contexto de utilização do sistema para a emissão do documento de Contrato.

O próximo diagrama de casos de uso desenvolvido é apresentado na figura 4.10.

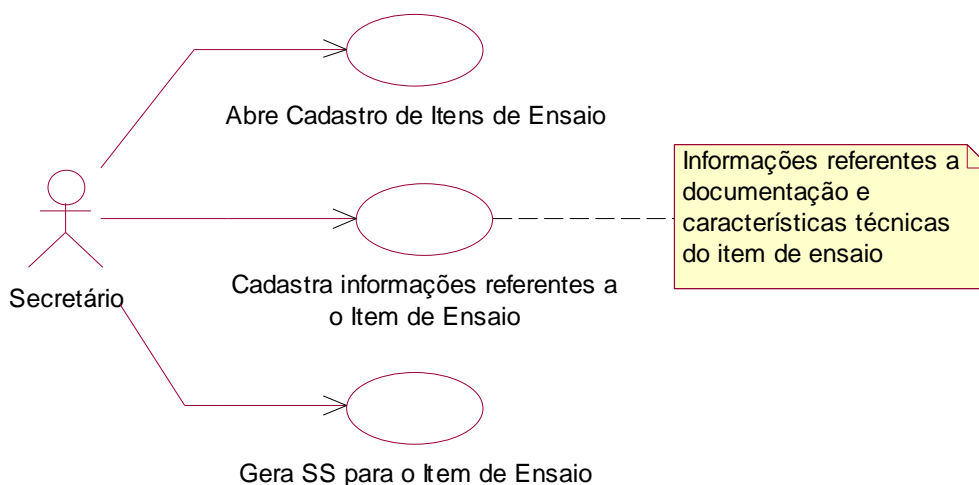


Figura 4.10 – Diagrama de Casos de Uso 'Refinamento do Caso de Uso Abre SS para o Item Recebido'.

O diagrama da figura 4.10 é um refinamento do caso de uso 'Abre SS para o Item Recebido' apresentado no diagrama de casos de uso da figura 4.4.

Da mesma forma que os diagramas de casos de uso desenvolvidos anteriormente nesta fase, este diagrama mostra o comportamento do usuário dentro do contexto de utilização do sistema.

O último diagrama de casos de uso desenvolvido nesta fase é mostrado na figura 4.11.

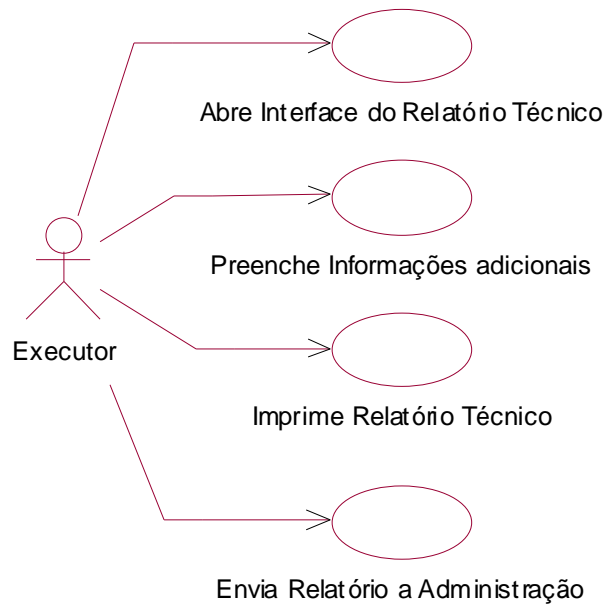


Figura 4.11 – Diagrama de Casos de Uso 'Refinamento do Caso de Uso Emitir Relatório Técnico'.

O diagrama da figura 4.11 é um refinamento do caso de uso 'Emitir Relatório Técnico' apresentado na figura 4.2.

Com a criação deste diagrama concluímos o desenvolvimento do modelo de casos de uso na fase de elaboração.

4.3.2.1.2 - Fluxo de Análise

O fluxo de análise da fase de elaboração tem como objetivo complementar o modelo de análise, inicialmente desenvolvido na fase de concepção, através da identificação de classes e seus respectivos relacionamentos mediante o estudo dos requisitos remanescentes capturados e representados através de casos de uso no fluxo de trabalho anterior.

A análise de tais requisitos resultou na atualização do modelo de análise como pode ser visto no diagrama de classes apresentado na figura 4.12.

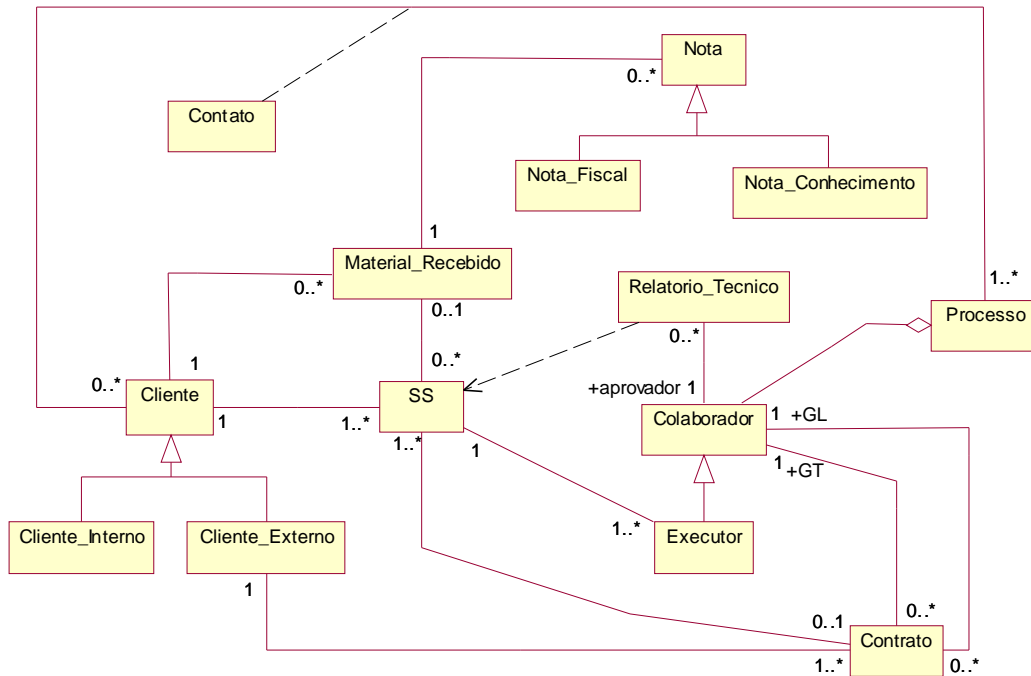


Figura 4.12 – Diagrama de Classes do Modelo de Análise. Fase de Elaboração – Interação #1.

O diagrama da figura 4.12 apresenta dois novos tipos de relacionamento, são a agregação e o atributo de ligação. O primeiro relaciona as classes ‘Processo’ e ‘Colaborador’, permitindo que se possa concluir que um processo é um agregado de colaboradores, ou seja, um colaborador *faz parte* de um processo. O segundo relaciona a classe ‘Contato’ com o relacionamento existente entre ‘Cliente’ e ‘Processo’, de forma que se possa afirmar que para cada relacionamento entre cliente e processo existe um contato específico.

A partir da atualização do modelo de análise, o diagrama de classes poderá ser analisado visando à identificação de pacotes que servirão de base para a criação de subsistemas no fluxo de projeto. A criação de pacotes e subsistemas será realizada durante a segunda interação da fase de elaboração.

4.3.2.1.3 - Fluxo de Projeto

O fluxo de projeto da fase de elaboração é responsável pela atualização do modelo de projeto através da complementação do diagrama de classes

lógico criado na fase de concepção. Esta atividade é realizada baseando-se no modelo de análise do fluxo anterior.

O diagrama de classes do modelo de projeto pode ser visto na figura 4.13.

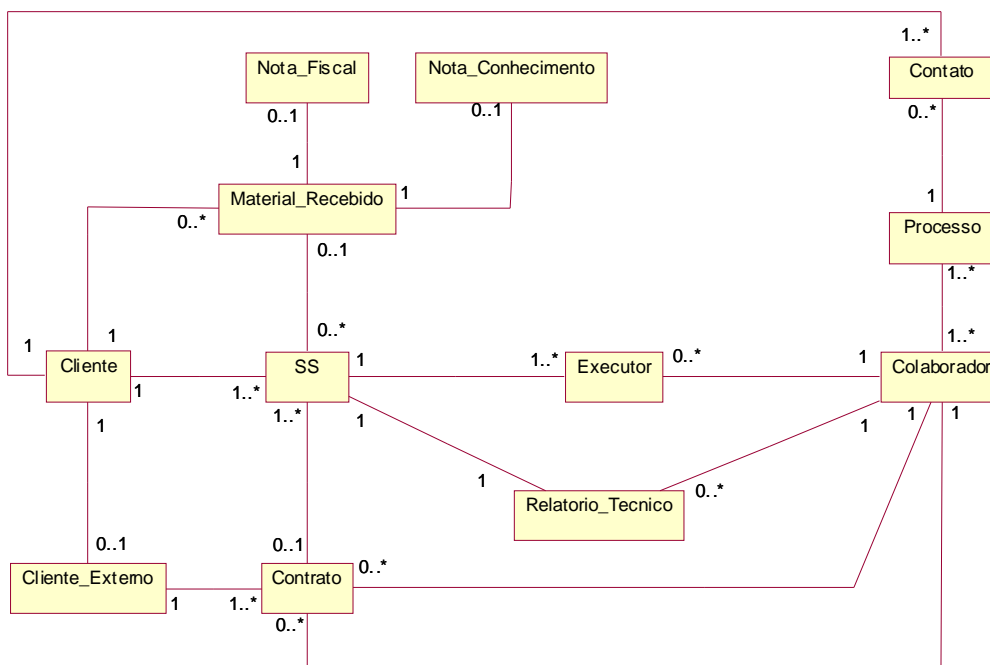


Figura 4.13 – Diagrama de Classes do Modelo de Projeto.
Fase de Elaboração – Interação #1.

Da mesma forma que na fase de concepção, o diagrama de classes do fluxo de projeto não apresenta os relacionamentos conceituais do modelo de análise, o que pode ser exemplificado pela substituição do atributo de ligação e da agregação por associações. Isto não compromete o modelo de projeto pelo fato das multiplicidades existentes nas associações garantirem a integridade dos relacionamentos.

A partir da criação do diagrama da figura 4.13 podemos dar início a segunda interação da fase de elaboração, delegando, neste caso específico, toda a tarefa de implementação à fase de construção.

4.3.2.2 – Interação #2

4.3.2.2.1 - Fluxo de Requisitos

O fluxo de requisitos durante a segunda interação da fase de elaboração não sofrerá alterações tendo em vista que o seu trabalho durante a primeira interação alcançou o objetivo da fase, no que diz respeito à identificação dos requisitos remanescentes.

4.3.2.2.2 - Fluxo de Análise

O papel do fluxo de análise durante esta segunda interação consiste, entre outras tarefas, na análise dos requisitos do modelo de casos de uso objetivando a coleta de informações necessárias para a definição dos atributos das classes do modelo de análise.

O resultado deste trabalho pode ser visto na figura 4.14.

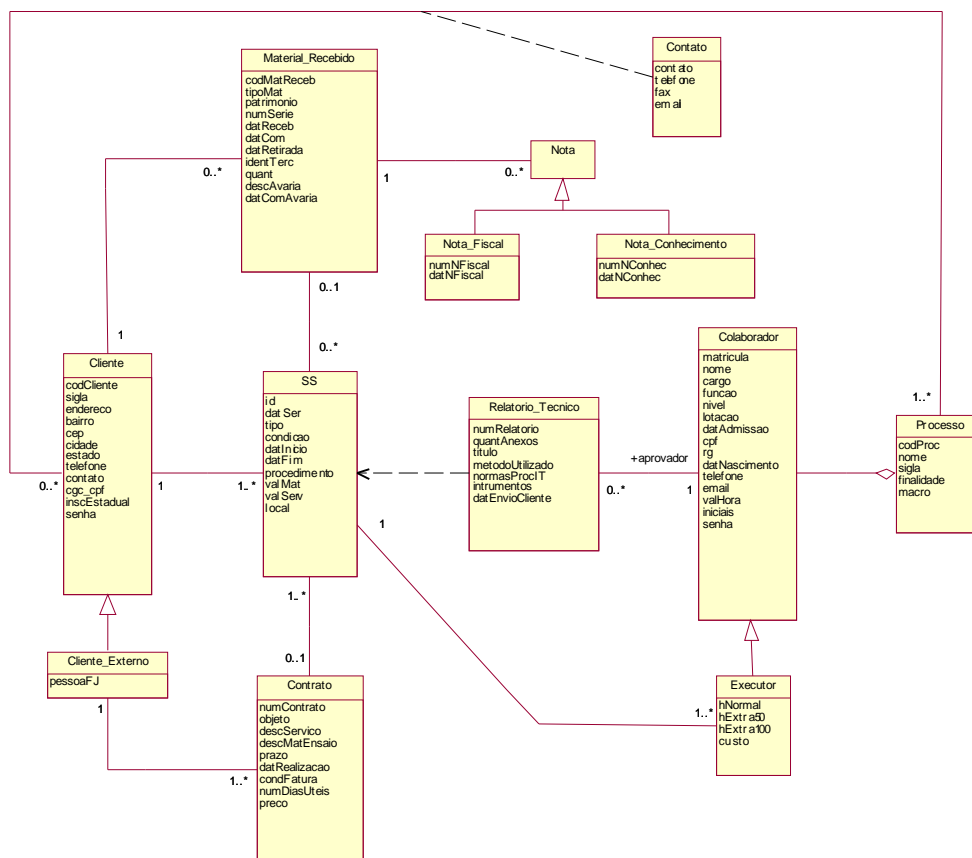


Figura 4.14 – Diagrama de Classes do Modelo de Análise. Fase de Elaboração – Interação #2.

Outra tarefa do fluxo de análise durante esta segunda interação é a identificação de pacotes através da análise do diagrama de classes da figura 4.14. Com base neste diagrama puderam ser identificados os pacotes 'Material_Recebido', 'SS' e 'Cliente'. Os mesmos correspondem, respectivamente, as figuras 4.15, 4.16 e 4.17.

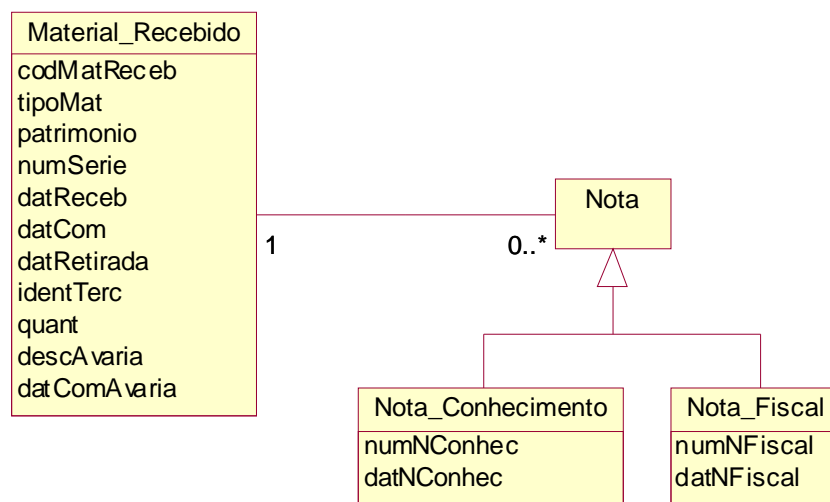


Figura 4.15 – Pacote Material_Recebido.

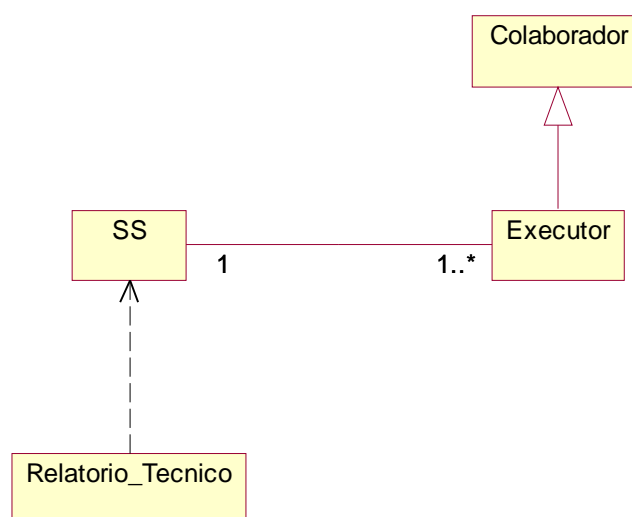


Figura 4.16 – Pacote SS.

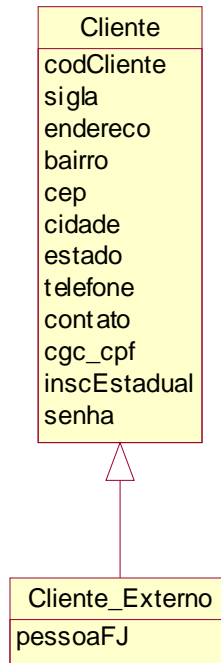


Figura 4.17 – Pacote Cliente.

A identificação destes pacotes servirá de base para a identificação dos subsistemas no fluxo de projeto.

4.3.2.2.3 - Fluxo de Projeto

Uma das tarefas deste fluxo de projeto consiste na modelagem de um diagrama de classes lógico a partir do diagrama de classes conceitual resultado do fluxo de análise.

O diagrama de classes lógico do modelo de projeto pode ser visto na figura 4.18.

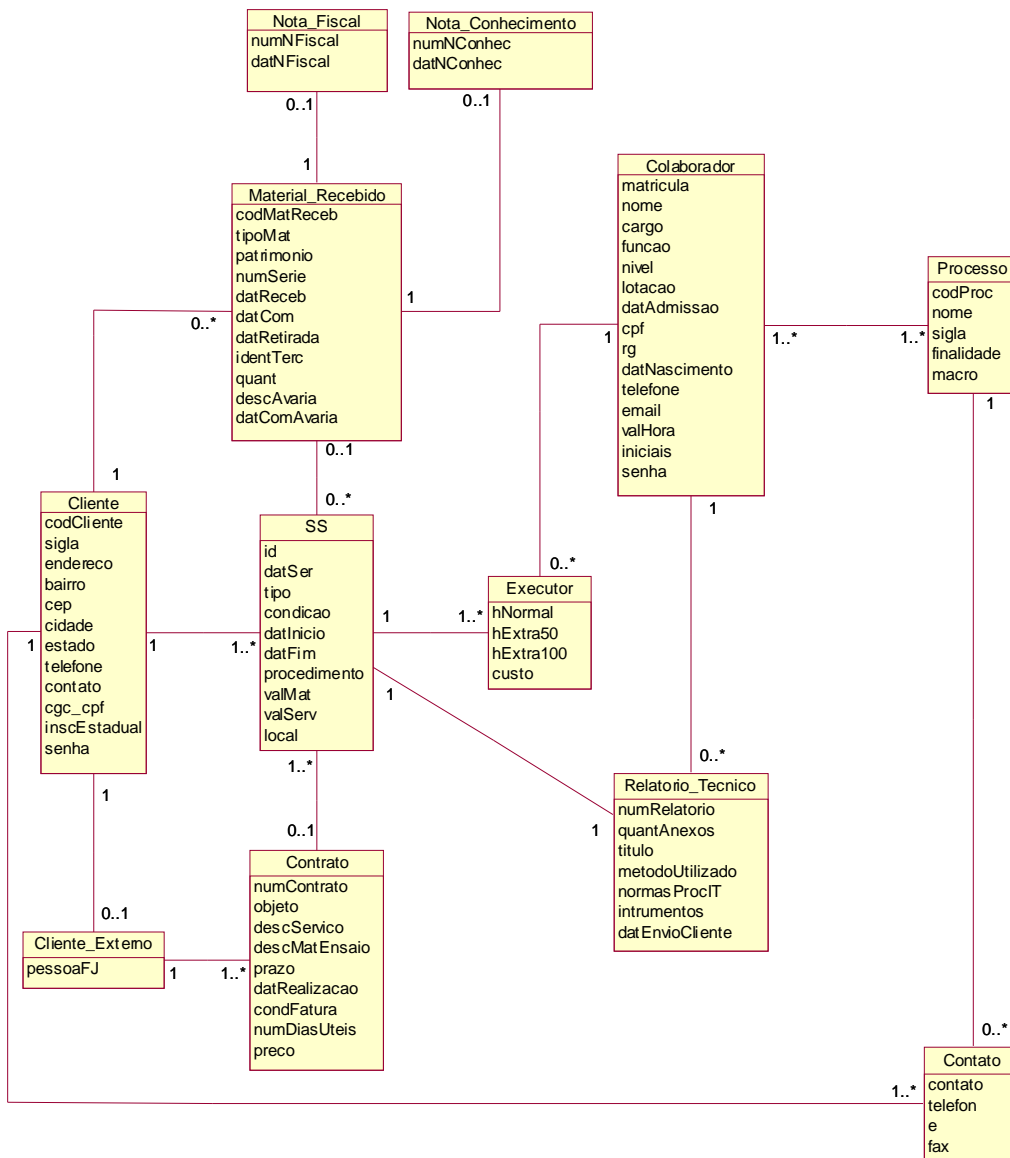


Figura 4.18 – Diagrama de Classes do Modelo de Projeto.
Fase de Elaboração – Interação #2.

Após a criação de um diagrama de classes lógico e da definição dos atributos de cada classe, tarefas como a criação de diagramas de interação, diagramas de gráficos de estados e diagramas de atividades podem ser realizadas.

A criação de diagramas de interação foi designada à fase de construção, com o propósito de modelar o comportamento do usuário diante das interfaces

gráficas do sistema e como estas interfaces gráficas interagem com o banco de dados.

No caso dos diagramas de gráfico de estados, exemplificamos na figura 4.19 como funciona a transição de estados de uma Solicitação de Serviço.

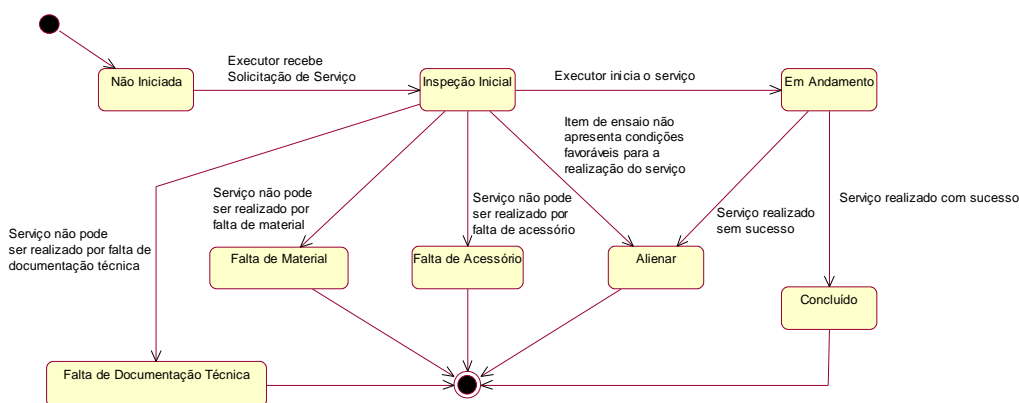


Figura 4.19 – Diagrama de Gráfico de Estados da Classe 'SS'.

Os estados do diagrama da figura 4.19 representam os valores que o atributo 'condicao' de um objeto da classe 'SS' pode assumir durante a execução do sistema.

Além de diagrama de gráfico de estados, o fluxo de projeto pode fazer uso de diagramas de atividades com o mesmo propósito de realizar a modelagem comportamental do sistema. Neste estudo de casos, foram criados diagramas de atividades que representam graficamente o fluxo das tarefas realizadas, tanto por usuários quanto pelo sistema, através do ciclo de vida de uma solicitação de serviço.

O primeiro diagrama de atividades apresentado neste estudo de casos, foi criado baseando-se no Procedimento de Recepção e Manuseio de Itens de Ensaio do Manual de Procedimentos do Lacen. Este diagrama representa o fluxo de tarefas realizadas por um colaborador competente, mediante a recepção de um item de ensaio enviado por um cliente.

Este procedimento antecede à abertura de uma solicitação de serviço, conforme pode ser visto na figura 4.20.

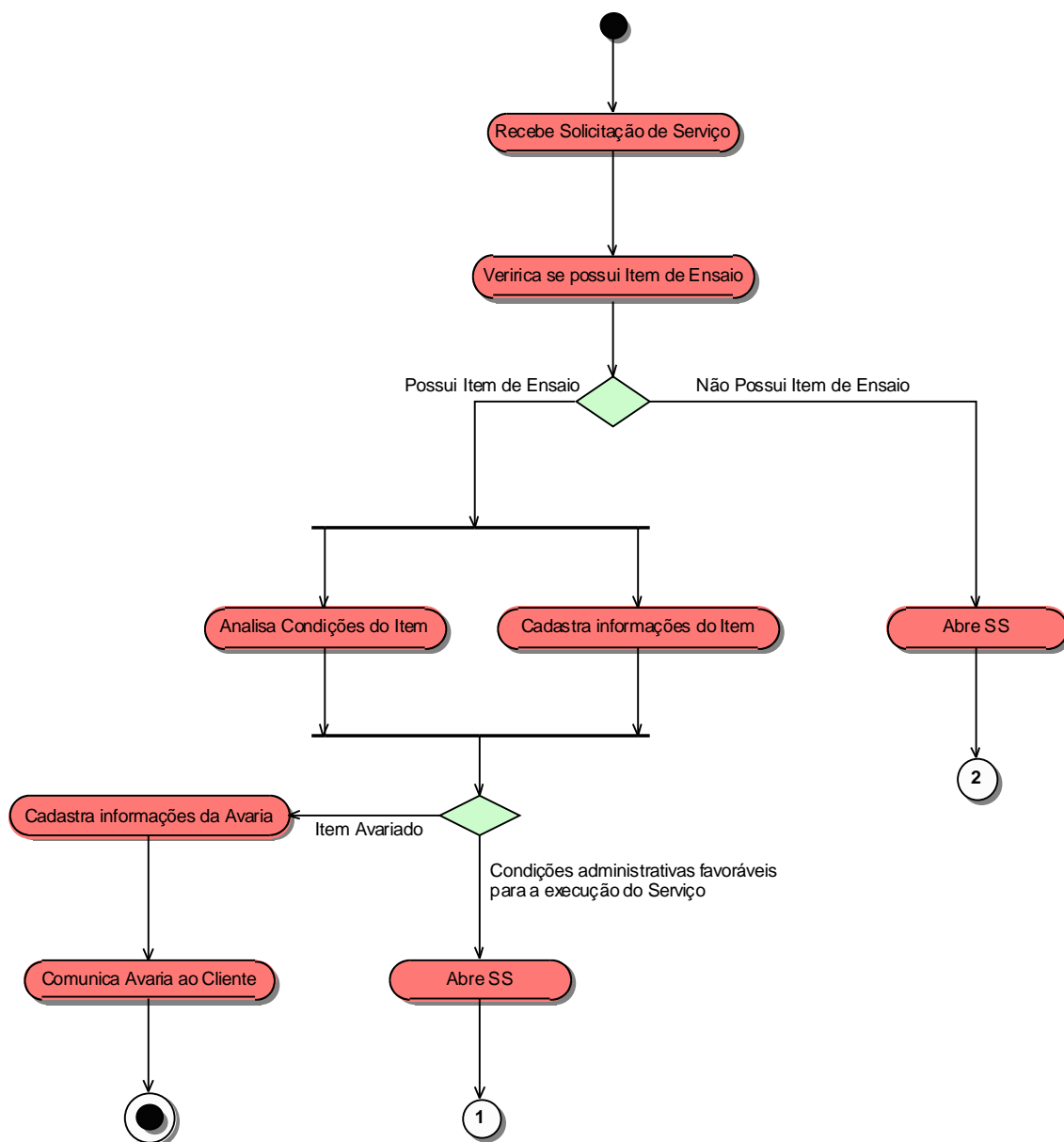


Figura 4.20 – Diagrama de Atividades 'PRIE'.

O diagrama da figura 4.20 possui um estado final, que é alcançado no caso de alguma avaria ter sido detectada no item de ensaio recebido, neste caso nenhuma 'SS' será aberta.

No caso das condições iniciais de um item de ensaio serem favoráveis ou mesmo no caso de não se possuir um item de ensaio como objeto do serviço, a solicitação de serviço pode ser aberta.

Baseado nestas considerações uma 'SS' é aberta de duas formas diferentes como indica a figura 4.21 e complementa a figura 4.22.

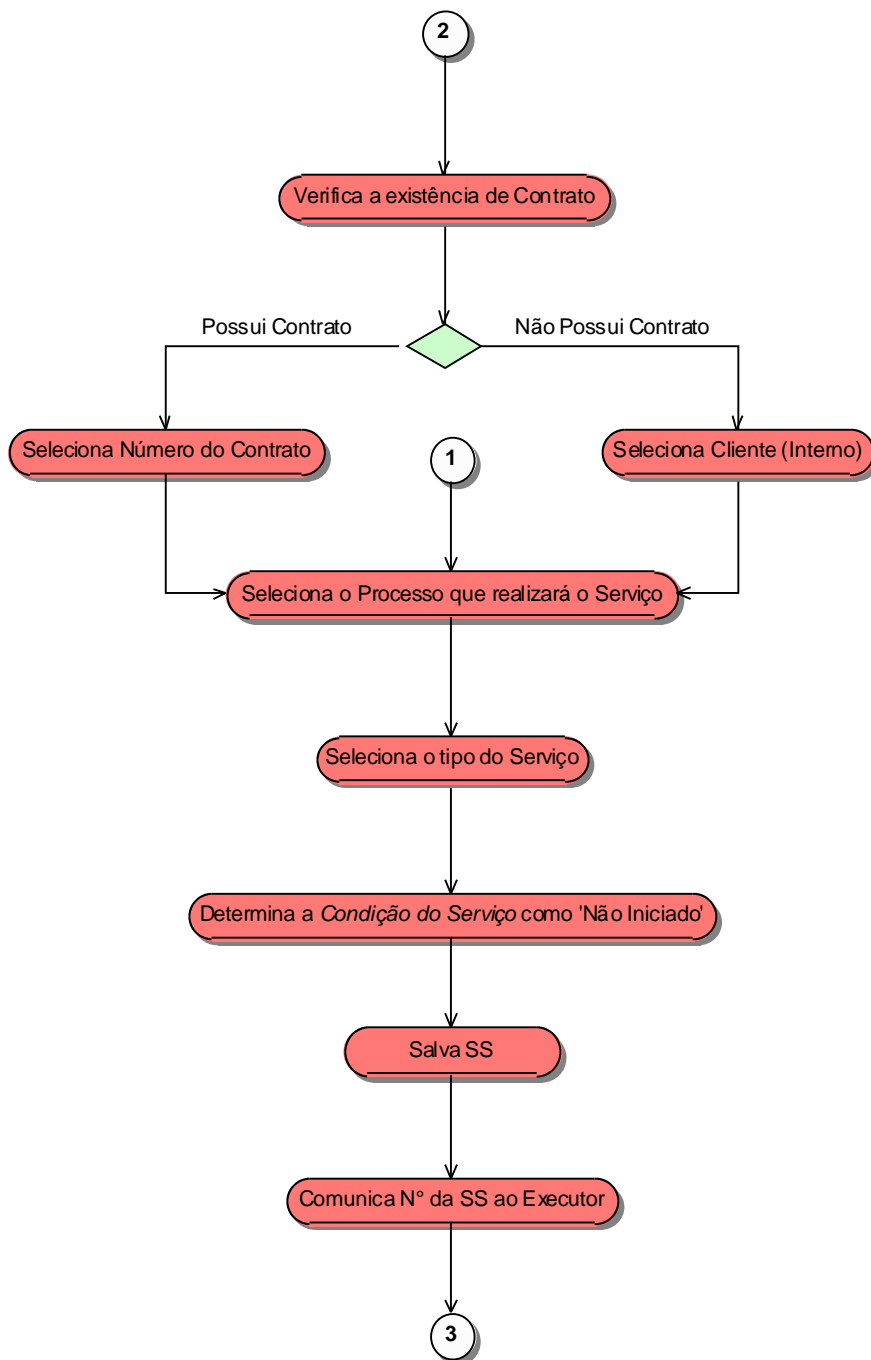


Figura 4.21 – Diagrama de Atividades 'Abre SS'.

Como pode ser visto no diagrama da figura 4.21, quando uma 'SS' é aberta considerando-se a existência de um item de ensaio, a primeira atividade a ser realizada é a definição do processo que realizará o serviço dentro do sistema de Solicitação de Serviço. O estado #1 aponta para esta atividade, pelo fato das atividades antecedentes, como definição do número do contrato e

cliente, já terem sido realizadas implicitamente pela atividade 'Cadastra Informações do Item' do diagrama de atividades da figura 4.20.

No caso da 'SS' ser aberta sem um item de ensaio, as atividades de definição de número de contrato e/ou cliente devem ser realizadas antes da definição do processo responsável pela realização do serviço, conforme indica o estado #2.

Perceba que neste último caso, se existir um contrato, o usuário não precisa especificar o cliente, tendo em vista que através da definição do número do contrato, o sistema conseguirá distinguir qual o cliente correspondente. O cliente só será especificado pelo usuário, caso não exista um contrato, o que indica que o cliente em questão é interno.

O próximo diagrama de atividades representa o fluxo de atividades realizadas pelo executor do serviço, no que diz respeito à complementação das informações de uma 'SS' aberta inicialmente pela administração. Este diagrama de atividades é mostrado na figura 4.22, continuação do diagrama da figura 4.21 a partir do estado #3.

O último diagrama de atividades representa o fluxo de atividades correspondentes a emissão de um Relatório Técnico após a conclusão de uma 'SS'. Este diagrama de atividades é mostrado na figura 4.23, continuação do diagrama da figura 4.22 a partir do estado #4.

Perceba que no diagrama da figura 4.23 o estado final é utilizado após a atividade 'Salva Registro', o que indica não só a finalização deste diagrama, mas também de todo o ciclo de vida de uma Solicitação de Serviço representado pelo conjunto dos quatro diagramas de atividades apresentados neste estudo de casos.

A apresentação destes quatro diagramas de atividades encerra o fluxo de projeto da fase de elaboração, o que nos permite direcionar o estudo de casos à fase seguinte, a fase de construção.

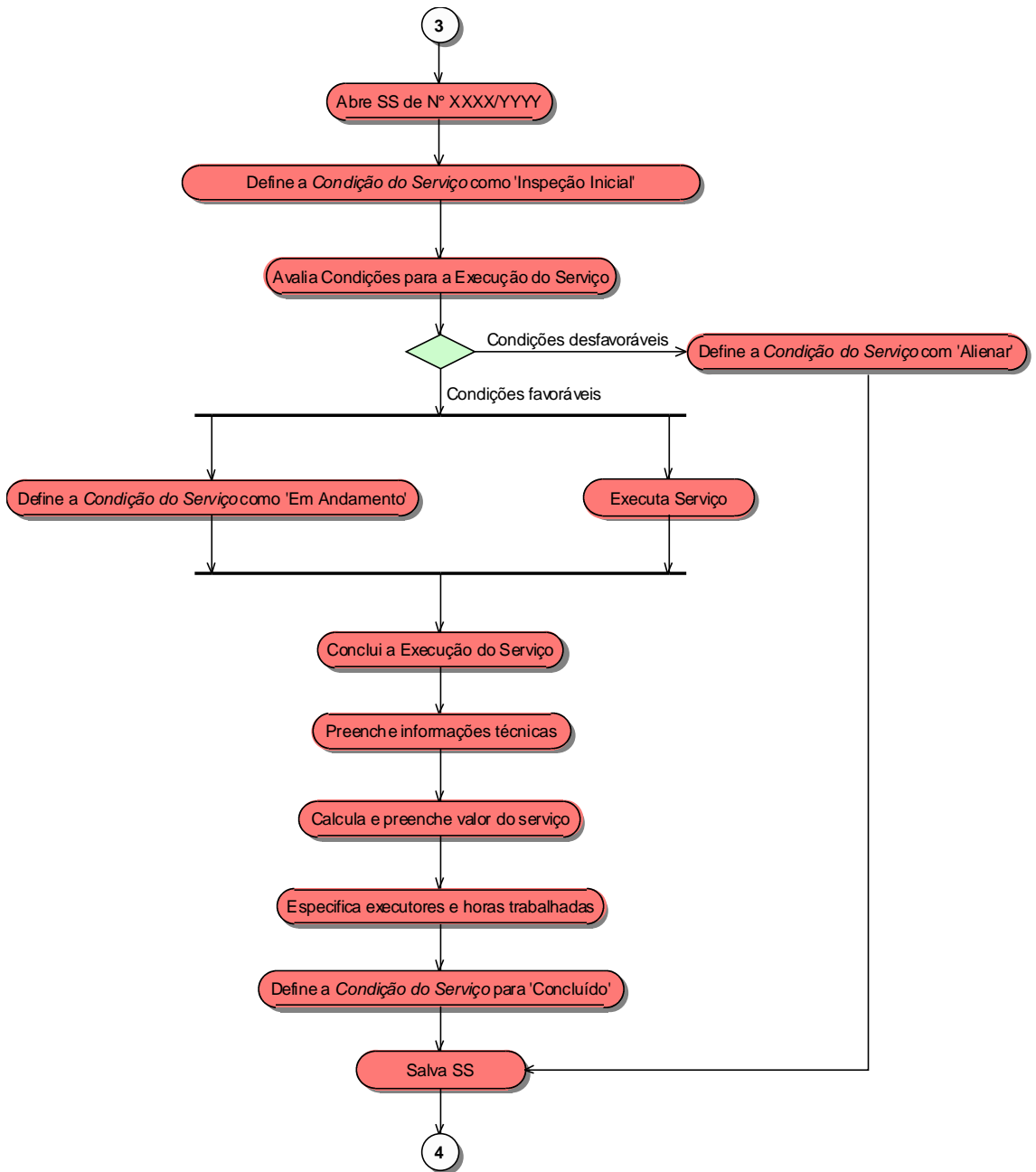


Figura 4.22 – Diagrama de Atividades ‘Conclui SS’.

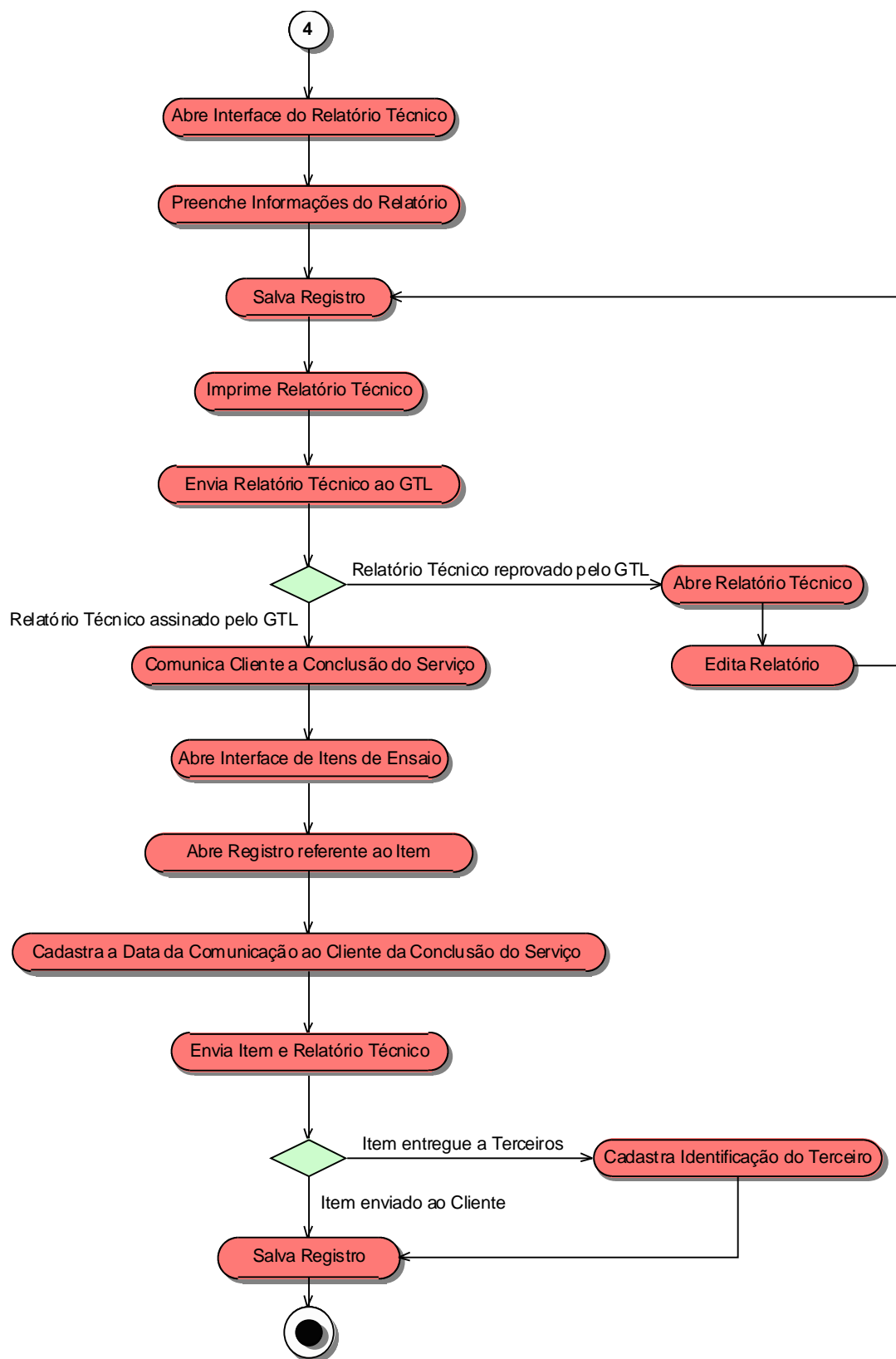


Figura 4.23 – Diagrama de Atividades 'Emitir Relatório Técnico'.

4.3.3 – Fase de Construção

O objetivo da fase de construção é produzir a versão beta do produto que está sendo desenvolvido, com base na arquitetura executável resultante da fase de elaboração.

Neste estudo de casos, algumas modificações são realizadas no modelo de projeto resultante da fase de elaboração, porém o maior trabalho da fase de construção está concentrado, de fato, no fluxo de implementação.

A fase de construção deste estudo de casos é desenvolvida através de uma única interação.

4.3.3.1 – Interação #1

4.3.3.1.1 - Fluxo de Requisitos

O fluxo de requisitos da fase de construção não sofrerá alterações, da mesma forma que na segunda interação da fase de elaboração, tendo em vista que todos os requisitos necessários foram identificados durante a fase anterior.

4.3.3.1.2 - Fluxo de Análise

O fluxo de análise da fase de construção não é executado pelo fato da estrutura do modelo de análise não ter sido preservada após a fase de elaboração, visando à diminuição de custos e o melhor aproveitamento de prazos como já foi discutido. Desta forma, qualquer eventual alteração na estrutura da arquitetura do sistema é de responsabilidade total do fluxo de projeto desta fase.

4.3.3.1.3 - Fluxo de Projeto

O papel do fluxo de projeto desta fase consiste, entre outras tarefas, na criação de diagramas de seqüência baseados em casos de uso identificados nas fases anteriores. Estes diagramas representam o comportamento do

usuário, caracterizado por um ator, diante das interfaces gráficas do sistema e como este sistema interage com as informações armazenadas em resposta às requisições do usuário.

Os diagramas de seqüência podem ser vistos nas figuras 4.24, 4.25, 4.26 e 4.27.

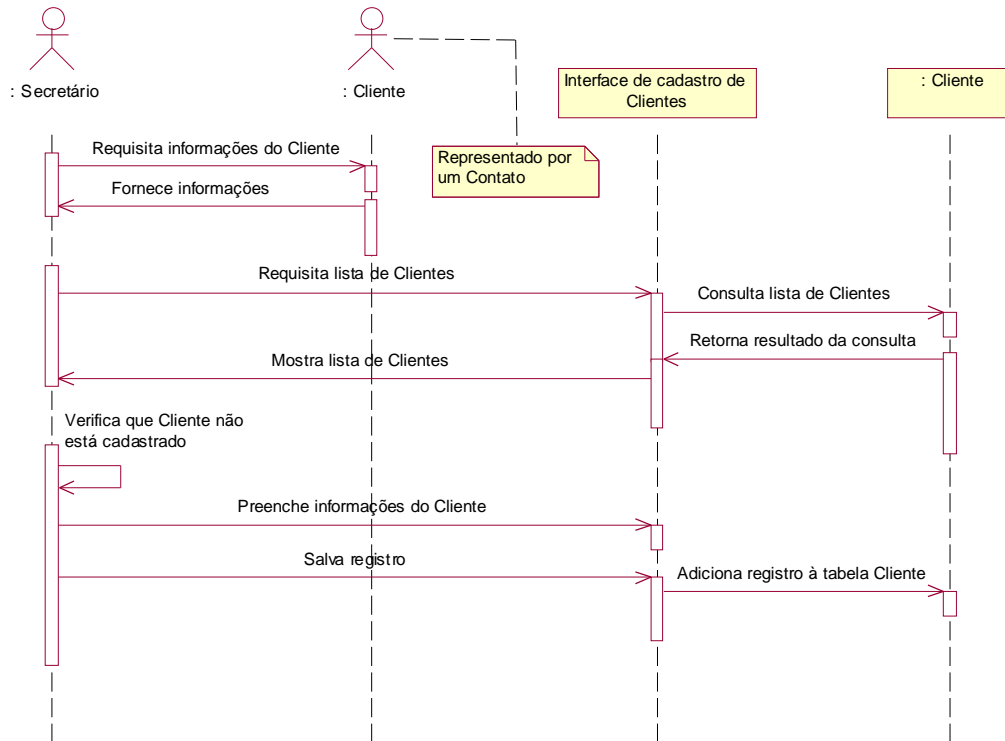


Figura 4.24 – Diagrama de Seqüências para o Caso de Uso 'Coleta Informações do Cliente'.

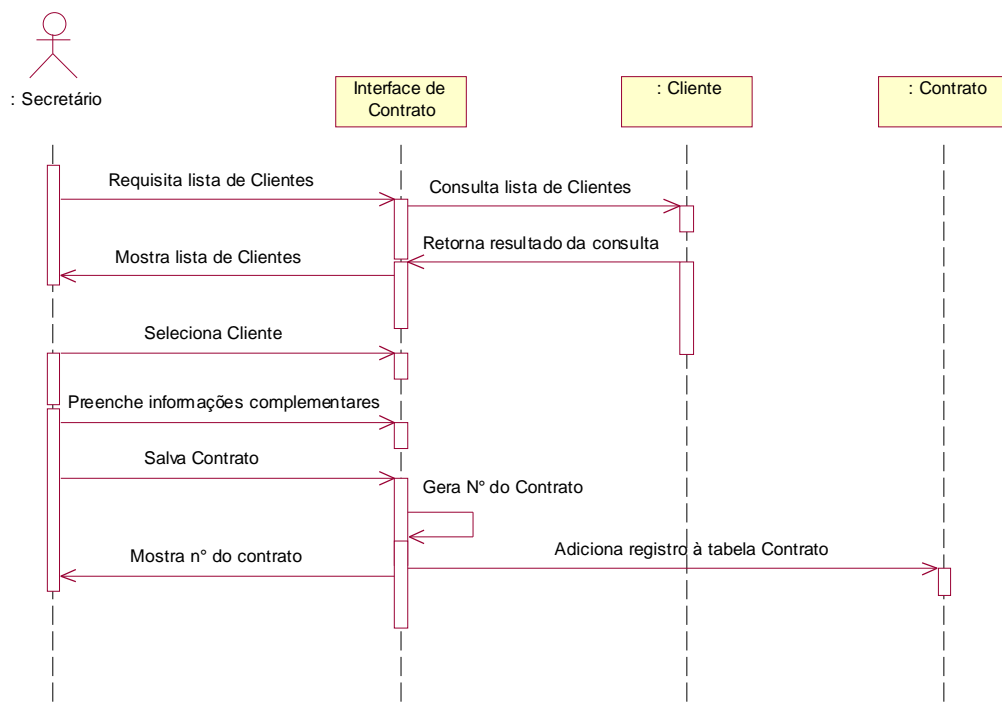


Figura 4.25 – Diagrama de Seqüências para o Caso de Uso 'Emite o documento Contrato para a Execução do Serviço'.

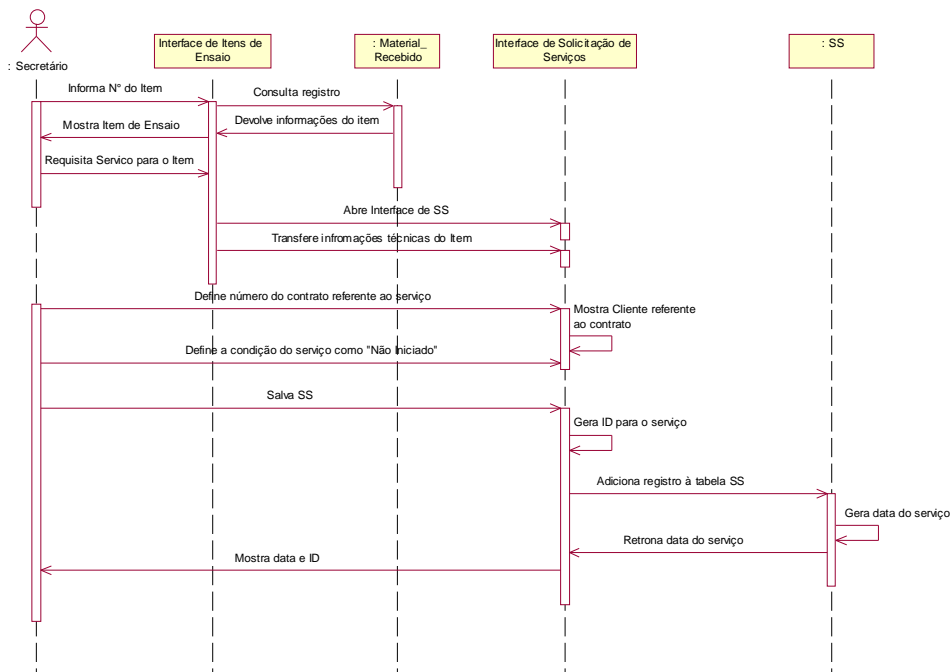


Figura 4.26 – Diagrama de Seqüências para o Caso de Uso 'Abre SS para o Item Recebido'.

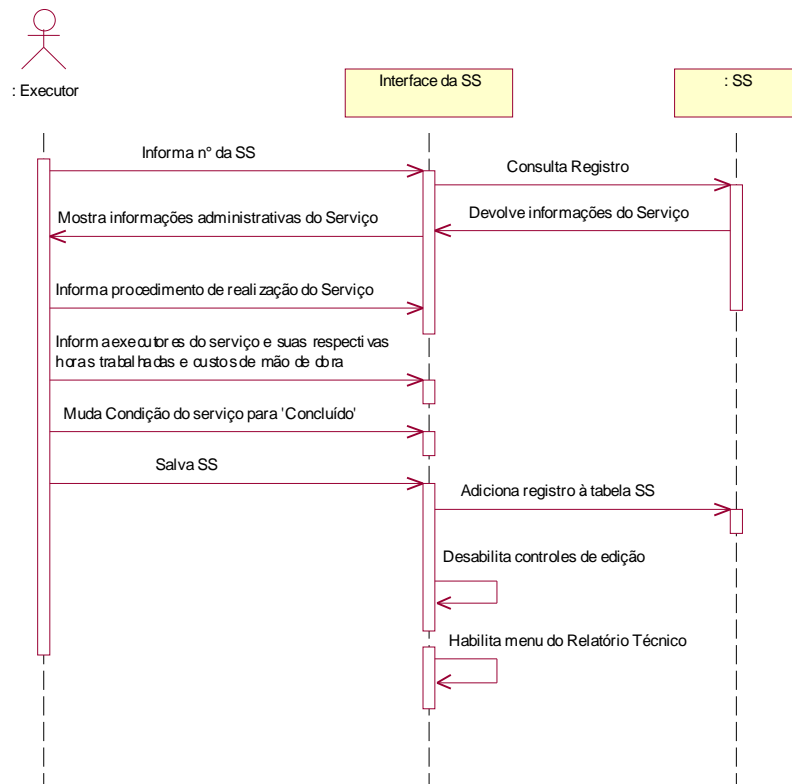


Figura 4.27 – Diagrama de Seqüências para o Caso de Uso 'Conclui SS'.

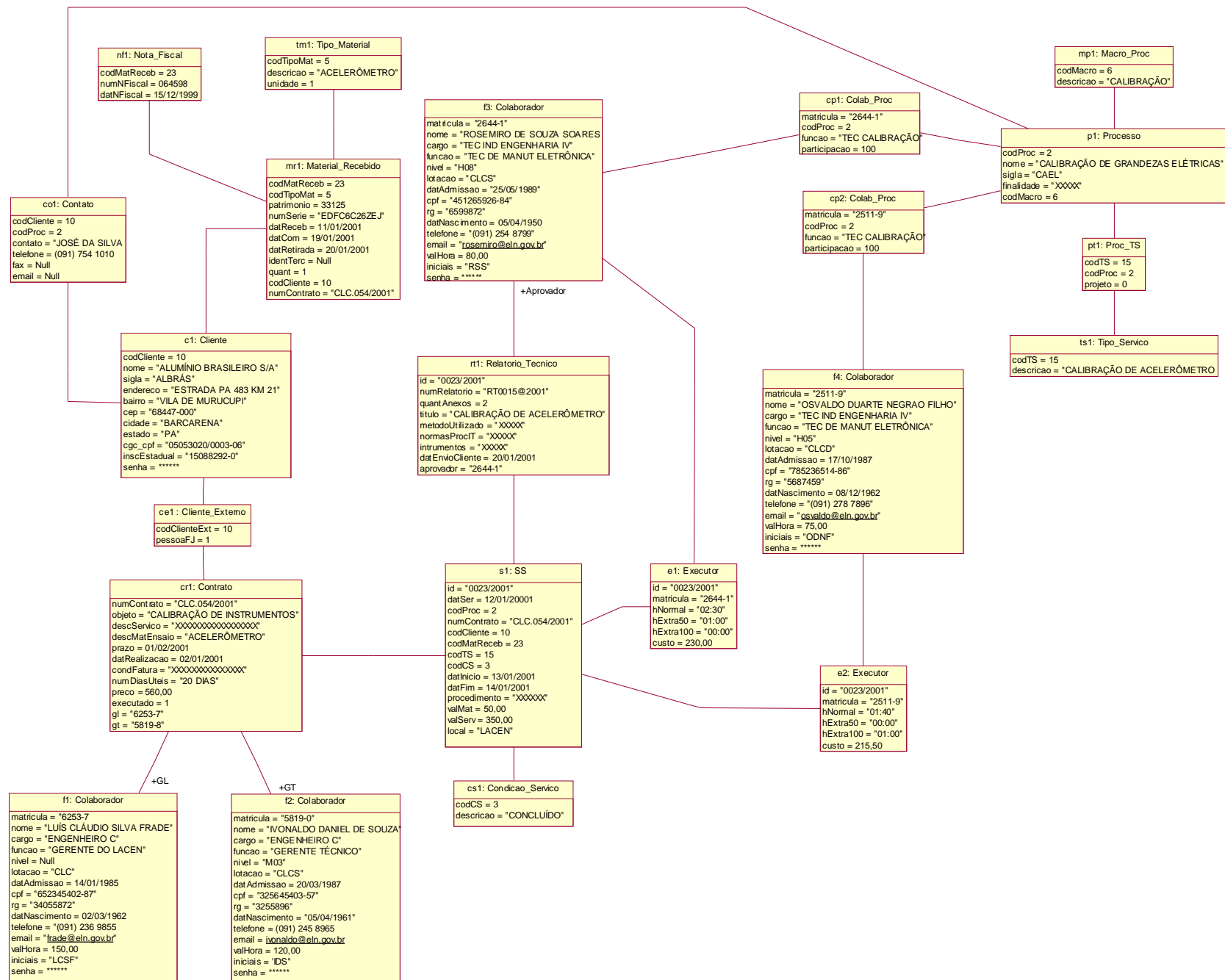


Figura 4.29 – Diagrama de Objetos para o Modelo Físico do Banco de Dados

Tendo em vista que o sistema não será implementado em uma linguagem de programação orientada a objetos, os objetos representados na figura 4.29 são na realidade registros, ou seja, linhas de tabelas do banco de dados criadas em determinado ponto no tempo de execução do sistema.

Para especificar as entidades de um modelo físico de banco de dados é necessária a utilização de uma notação especial. Pelo fato da UML não fazer nenhuma referência a notações desta espécie, este estudo de casos utilizará a notação da análise essencial para dicionário de dados.

A tabela abaixo faz alguns esclarecimentos a respeito desta notação.

Símbolo	Significado
=	é composto de
+	e
()	opcional (pode estar presente ou ausente)
{ }	iteração
[]	escolha em uma das alternativas
**	comentário
@	identificador (chave) em um depósito
/	separa opções alternativas na construção []
#	número

Tabela 4.1 – Notação para o Dicionário de Dados.

Para exemplificar esta notação, serão descritas abaixo duas entidades do modelo da figura 4.28, 'SS' e 'Executor'.

SS = {serviços}

serviços = * Informações administrativas e técnicas dos serviços realizados pelo Laboratório Central. *

@Id + datSer + datInicio + datFim + procedimento + valMat + valServ + local. *

Id = * Identificador do serviço, campo texto de formato 'xxxx/yyyy', sendo que xxxx representa um número inteiro seqüencial e yyyy o ano em que ocorreu a solicitação. *

valMat = * Custo do material utilizado para a execução do serviço, em reais. *

valServ = * Custo em reais do serviço realizado. Se o cliente for interno, o valServ é calculado com base na fórmula $valServ = \sum Executor.custo + valMat$. Se o cliente for externo, o valServ é definido pelo executor responsável pelo serviço. *

procedimento = * Descreve o procedimento de realização do serviço. *

local = * Local onde o serviço foi realizado. *

EXECUTOR = {executor}

executor = * Colaborador responsável pela execução do serviço *
 @Matricula + @Id + hNormal + hExtra50 + hExtra100 + custo

Matricula = * Identificador de um colaborador no formato '#####-#' *

hNormal = * Tempo em minutos correspondente a execução do serviço durante o período de expediente normal. *

hExtra50 = * Tempo em minutos correspondente a execução do serviço após o expediente normal (dias úteis). *

hExtra100 = * Tempo em minutos correspondente a execução do serviço aos sábados e domingos. *

custo = * Custo do serviço. Corresponde à fórmula:
 $custo = ((hNormal + 1,5 \times hExtra50 + 2 \times hExtra100) / 60) \times Colaborador.valHora$. *

valHora = * Campo da tabela 'Colaborador' que representa o preço de 1 hora trabalhada do colaborador, em reais. *

A partir da criação do modelo físico de banco de dados o fluxo de implementação o processo de desenvolvimento inicia o fluxo de implementação da fase de construção.

4.3.3.1.4 - Fluxo de Implementação

Este é o fluxo de trabalho no qual o maior esforço da fase de construção é empregado. Este esforço consiste na implementação efetiva do sistema, tendo como base o modelo de projeto revisto e atualizado durante o fluxo anterior.

O resultado deste fluxo é submetido, posteriormente, ao fluxo de teste que disponibilizará a versão operacional inicial do sistema, representado 100% dos casos de uso.

Este estudo de casos mostra algumas das telas do SIGLacen, de forma a representar o resultado do trabalho exercido nesta etapa do processo.

Comentários a respeito do código fonte do sistema não se fazem necessários neste estudo de casos, tendo em vista que abordagens à linguagem de programação utilizada (Visual Basic 6) fogem do escopo deste trabalho.

As interfaces gráficas são apresentadas em uma ordem seqüencial que representa o ciclo de vida de uma solicitação de serviços, o que foi representado pelos diagramas de atividades criados durante a fase de elaboração.

A primeira tela apresenta o menu de acesso ao subsistema de gestão de serviços e instrumentos (SGSI) a partir da tela principal do SIGLacen. Esta interface gráfica pode ser vista na figura 4.30.



Figura 4.30 – Tela Principal – Subsistema SGSI.

Após se identificar através de sua matrícula e senha, o usuário possui acesso aos recursos do sistema de sua competência. Neste caso específico, um colaborador do processo de administração acessa o SIGLacen com o intuito de cadastrar um item de ensaio recebido para análise. Desta forma, a interface gráfica de cadastro de itens de ensaio deverá ser acessada através do botão SGSI.

A tela para cadastro de itens de ensaio pode ser vista na figura 4.31.

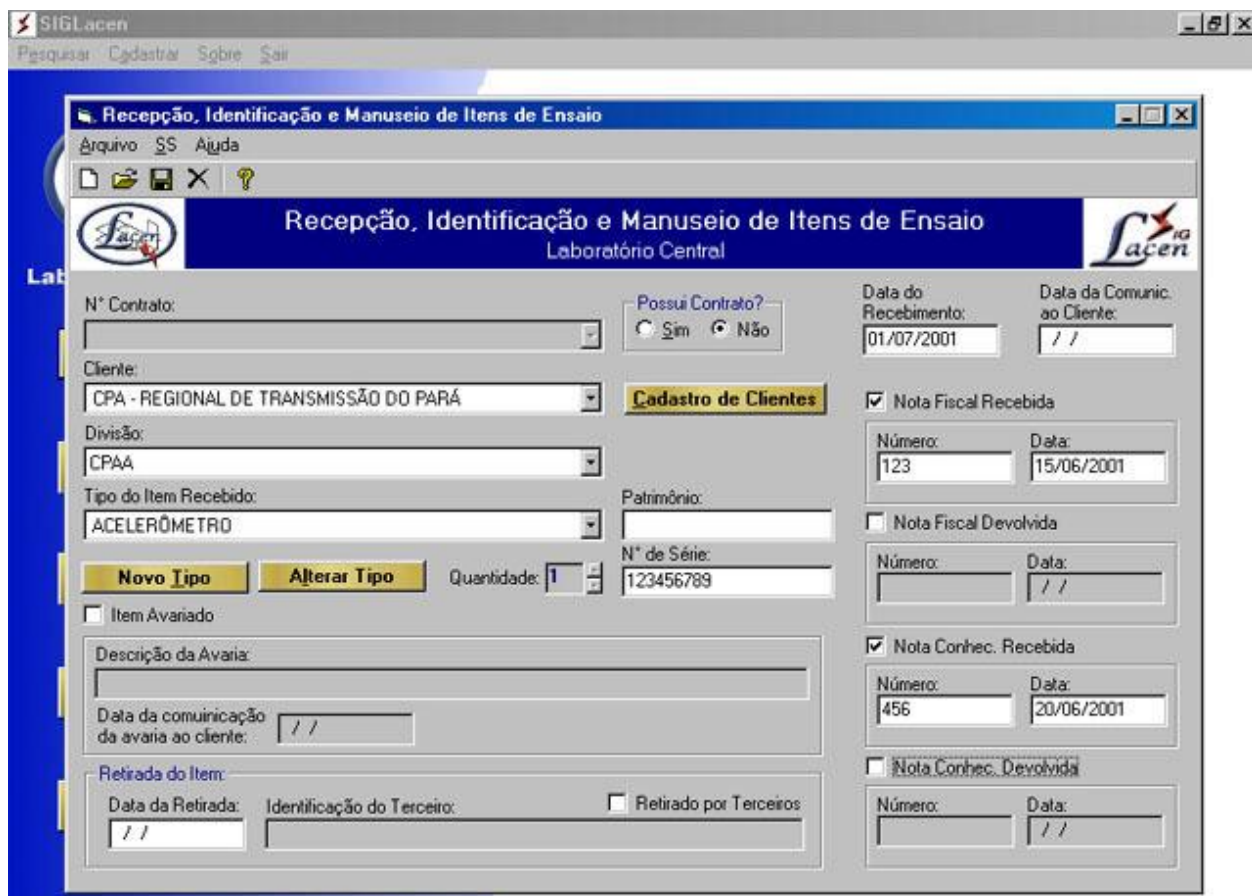


Figura 4.31 – Interface Gráfica para Recepção, identificação e manuseio de Itens de Ensaio.

Neste momento, apenas as informações referentes à recepção do item são cadastradas, possibilitando que uma 'SS' seja aberta tendo o item cadastrado como objeto do serviço. Após a conclusão da SS o colaborador do processo de administração deverá retornar a este cadastro para complementar as informações de devolução do item ao cliente.

A primeira tela do cadastro de solicitações de serviços pode ser vista na figura 4.32.

*Figura 4.32 – Interface Gráfica para Solicitação de Serviço
Cadastro das informações administrativas do serviço.*

Após o cadastro do item recebido, o colaborador do processo de administração gera uma solicitação de serviços que será aberta e complementada posteriormente pelo colaborador responsável pela execução deste serviço. Inicialmente, apenas as informações administrativas são cadastradas como pode ser visto na figura 4.32.

A seção correspondente às informações técnicas do serviço é mostrada na figura 4.33.

Solicitação de Serviço (SS)
Laboratório Central

Administrativo | Executores

Descrição do Serviço (Procedimento):
Após detectarmos incoerência nas medições do certificado em referência, devido as configurações dos filtros, foi comunicado ao cliente através da CI CLC-097/2000 solicitando nova medição e gerando a revisão no certificado de calibração. veja certificado de calibração CC05712000.

Data de Início: 07/08/2001 | Data de Término: 08/08/2001

Localização do Equipamento/Serviço: LACEN

Custo do Material (R\$): 55,00 | Preço do Serviço (R\$): 264,00

Selecionar por: Nome Matrícula

Colaborador: PAULO THADEO DE ANDRADE SILVA

H Normal: 00:00 | 50%: 00:00 | 100%: 00:00

Matrícula	Nome	H.Normal	H.Ext 50%	H.Ext 100%
1918-6	LUIZ SERRA DE ALMEIDA	01:00	01:00	00:00
6349-5	PAULO THADEO DE ANDRADE SILVA	02:00	00:00	00:00

Adicionar | Remover

Figura 4.33 – Interface Gráfica para Solicitação de Serviço
Cadastro das informações técnicas do serviço.

O executor responsável tem conhecimento do serviço a ser realizado através do número de identificação do mesmo informado pela administração.

Após a conclusão do serviço, o executor complementa o registro com informações técnicas, de custos, datas e horas utilizadas para a execução do mesmo, e define a condição do serviço como 'Concluído', o que impossibilita qualquer alteração futura neste 'SS'.

A partir daí, o relatório da 'SS' pode ser impresso para que a conclusão do serviço possa ser formalizada através da assinatura do Líder do Processo correspondente.

O relatório da 'SS' pode ser visto na figura 4.34.

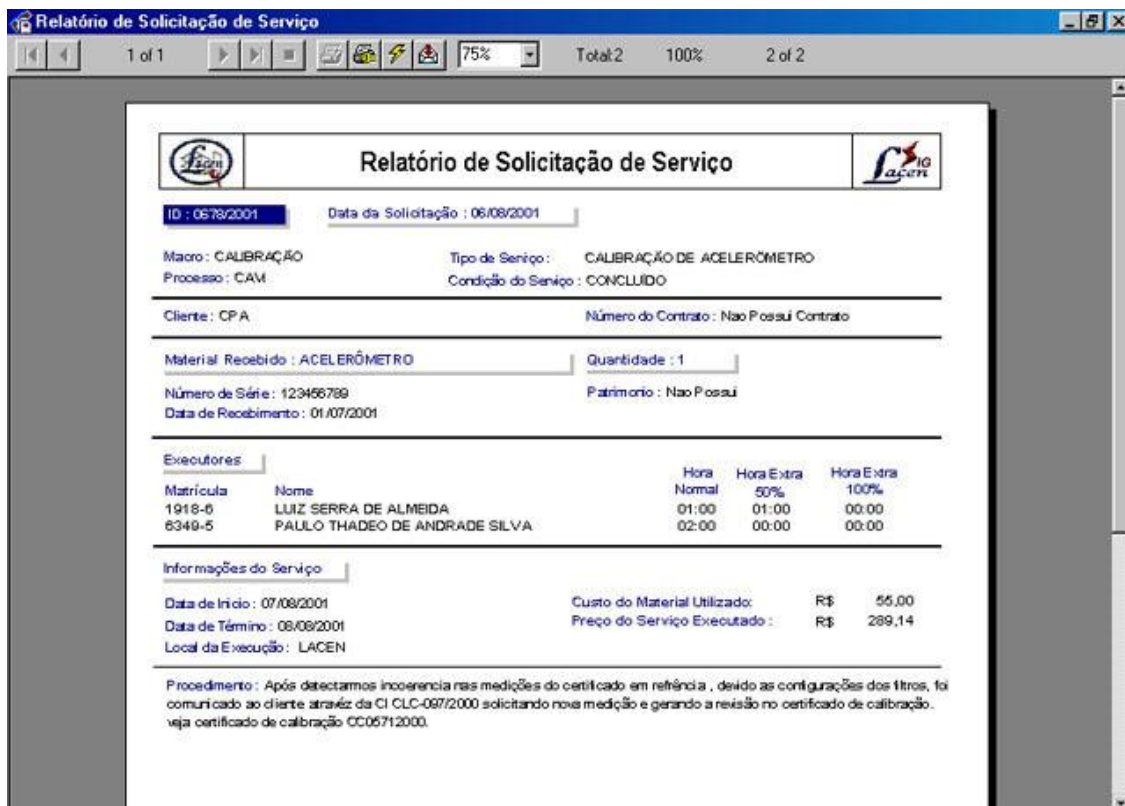


Figura 4.34 – Relatório da ‘SS’.

Após a devolução do item ensaio à administração, o colaborador responsável pela recepção do mesmo deverá cadastrar as informações de devolução do item no cadastro de itens de ensaio.

A interface gráfica correspondente ao cadastro das informações de devolução do item de ensaio pode ser vista na figura 4.35.

Figura 4.35 – Cadastro das informações de devolução do Item de Ensaio.

O procedimento de cadastro das informações de devolução do item de ensaio caracterizam o fim do ciclo de vida de uma Solicitação de Serviço.

Após a conclusão do fluxo de implementação e teste da fase de construção, o processo avança para a fase de transição, onde a versão operacional inicial do sistema é disponibilizada à uma comunidade de usuários com o intuito identificar deficiências mínimas que passaram despercebidas pela fase de construção e possam ser corrigidas dentro da arquitetura existente.

Se for verificada a aceitação do sistema por parte dos usuários e se o mesmo alcançou os objetivos previamente estipulados, o processo de desenvolvimento estará concluído. Caso contrário um novo ciclo de desenvolvimento deverá ser iniciado, objetivando a correção das eventuais deficiências encontradas.

Capítulo 5 – Conclusão

As abordagens realizadas sobre a utilização da UML dentro do Processo Unificado foram suficientes para o desenvolvimento do estudo de casos do capítulo 4. Porém, muito ainda se pode aprender com a Metodologia Unificada, tendo em vista que a mesma possibilita o desenvolvimento de sistemas de pequeno à grande porte, em todos os tipos de domínios de problemas.

Uma prova da flexibilidade da utilização da UML dentro do Processo Unificado para a modelagem de sistemas é evidenciada no estudo de casos apresentado neste trabalho, considerando que o sistema estudado foi implementado em uma linguagem não orientada a objetos.

É óbvio que a escolha de uma linguagem orientada a objetos possibilita o melhor aproveitamento dos recursos da Metodologia Unificada, entretanto, considerando fatores que envolvem o desenvolvimento de sistemas como investimentos, custos, prazos, recursos humanos e materiais, nem sempre isto será possível. De qualquer forma, a escolha de uma ou outra linguagem de programação não impedirá que Processo Unificado alcance seu objetivo, desenvolver um sistema obedecendo a estimativas de custos e prazos, de forma eficaz e principalmente, padronizada.

Usar a UML e o Processo Unificado para a modelagem de sistemas é muito simples. Porém, apenas com uma certa experiência na utilização da Metodologia Unificada se conseguirá domina-la, compreendendo suas características mais sutis.

A UML e o Processo Unificado compõem uma metodologia moderna e eficaz, entretanto, a lacuna evidenciada pela ausência de um recurso padronizado para a especificação dos atributos de uma classe, precisa ser preenchida. De qualquer modo, o estudo de casos apresentado neste trabalho conseguiu suprir esta deficiência fazendo uso do Dicionário de Dados da análise essencial.

O estudo de casos deste trabalho abordou a modelagem de um sistema implementado em uma linguagem visual, concentrando-se nos fluxos de análise e projeto orientado a objetos do Processo Unificado. Por este motivo,

uma sugestão para um trabalho futuro seria a implementação de um sistema usando uma linguagem orientada a objetos com o intuito de obter o máximo proveito dos recursos da Metodologia Unificada.

Referências Bibliográficas

[1] JACOBSON, Ivar; BOOCH, Grady; RUMBAUGH, James; **UML Guia do Usuário**. Rio de Janeiro: Campus, 2000. 473 p.

[2] JACOBSON, Ivar; BOOCH, Grady; RUMBAUGH, James; **The Unified Software Development Process**. Massachusetts: Addison-Wesley, 1999. 463 p.

[3] **Rational – The e-development company**. [On-line]. Consultado em 15/01/2001. Disponível em : <http://www.rational.com/index.jsp>

[4] **Applying UML in The Unified Process**. [On-line]. Consultado em 17/01/2001. Disponível em : <http://www.jeckle.de/files/uniproc.pdf>

[5] **Processos em Engenharia de Software**. [On-line]. Consultado em 20/01/2001. Disponível em : <http://ead1.eee.ufmg.br/~renato/engsoft/Cap02.pdf>

[6] **The Unified Process**. [On-line]. Consultado em 20/02/2001. Disponível em : http://www.it.nuigalway.ie/~o_molloy/courses/ct321/Lecture3/sld012.htm

[7] **UML & Unified Process**. [On-line]. Consultado em 20/02/2001. Disponível em : <http://www.mitrais.com/pdf/rup.pdf>

[8] **A Metodologia Unificada**. [On-line]. Consultado em 14/03/2001. Disponível em: <http://www.pr.gov.br/celepar/celepar/batebyte/Internet-anteriores/1999/bb89/software.htm>

[9] **O Ciclo de Vida do Processo Unificado**. [On-line]. Consultado em 20/03/2001. Disponível em : <http://citsqs.cits.br/download.htm>