A FEDERAL DE ENGENHARIA DE ITAJUBÁ

Modelo para Desenvolvimento de Software Administrativo da EFEI

CLÁUDIO DAS NEVES FRANCO DE SÁ

ITAJUBÁ - Dezembro
2001

ESCOLA FEDERAL DE ENGENHARIA DE ITAJUBÁ - EFEI
INSTITUTO DE ENGENHARIA ELÉTRICA – IEE
DEPARTAMENTO DE ELETRÔNICA - DON

DISSERTAÇÃO:

MODELO PARA DESENVOLVIMENTO DE SOFTWARE ADMINISTRATIVO DA EFEI

Dissertação apresentada à Escola Federal de Engenharia de Itajubá para obtenção do Título de Mestre em Engenharia

Área de concentração:
Automação de Sistemas

Orientador:
Prof. Dr. Otávio Augusto Salgado Carpinteiro

Mestrando:
Cláudio das Neves Franco de Sá - 8523

Itajubá
Dezembro/2001

Sá, Cláudio das Neves Franco de

Projeto de Software: Uma Metodologia para o desenvolvimento de software Administrativo da EFEI, tendo como fonte do estudo o Sistema de Informação do Docente. Aplicação das tecnologias do Modelo Relacional e UML (Unified Modeling Language). Itajubá – MG, 2001.

Dissertação (Mestrado) – Escola Federal de Engenharia de Itajubá. Instituto de Engenharia Elétrica. Departamento de Eletrônica.

1. Projeto de Software 2.Especificações 3. Modelo Relacional 5.UML
6. Avaliação de Riscos 7. Teste de Software 8. Engenharia de Software.

AGRADECIMENTOS

Ao orientador Prof. Dr. Otávio Augusto Salgado Carpinteiro pela valiosa orientação.

Ao Prof. Dr. Ricardo Rhomberg Martins (UFRJ) e Prof. Dr. João Bosco Schuman Cunha (EFEI), pela participação na banca examinadora e valiosos observações e sugestões.

A todos os professores do Curso de Mestrado em Engenharia Elétrica da EFEI pelos valiosos conhecimentos repassados.

Um especial agradecimento à minha família.

A todos que colaboraram e me incentivaram.

SUMÁRIO

RESUMO

"ABSTRACT"

Página

1. Introdução 09
2. Tecnologias aplicadas no projeto 12

Técnicas de Coleta de Informações - Modelagem Lógica de Sistemas 12

Banco de Dados – Uml (Unified Modeling Language) 17

3. Desenvolvimento do Software – Etapas 29

Estratégia do desenvolvimento do projeto 29

3.1. Estudo Inicial – Coleta de Informações – Entrevistas 31

Modelo de Formulário de entrevista utilizado 31

3.2. Elaboração da Proposta de Desenvolvimento 33

3.3. Levantamento/Análise Preliminar de Requisitos(LPR) 43

3.3.1. Modelo Descritivo(Descrição Narrativa) 43

3.4. Requisitos do Sistema 43

3.5. Análise do Projeto Lógico de Banco de Dados 44

3.5.1. Desenv. do Modelo Conceitual Entidade-Relacionamento (DER) 44

3.5.1.1. DER – ferramenta Flow Charting 4 45

3.5.1.2. DER – formato Flow Charting4 46

3.5.1.3. DER – ferramenta Logic Works Erwin_Erx3.0 47

3.5.1.4. DER – formato Logic Works Erwin_Erx3.0 48

3.6. Projeto Físico

3.6.1. Modelo Relacional Normalizado 50

3.7. Requisitos do Sistema após a Modelagem Relacional 51

3.8. Elaboração do Dicionário de Dados - Entidades/Relacionamentos 52

3.9. Clientes do Sistema – Classes de Usuários 54

3.10. Funções do Sistema 54

3.11. Diagramas UML – Ferramenta: Rational Rose/C++ Demo 55

3.12. Cenários Use-Case 56

3.13. Diagrama Use-Case (Visão de Caso de Uso) 57

3.14. Cartões Cenários Use Cases 59

3.15. Classes Candidatas da Aplicação 64

3.16. Levantamento de Classes de Interfaces com o Usuário 65
3.17. Classes de Persistência 66
3.18. Lista de Responsabilidades da Aplicação 67
3.19. Cartões das Classes de Aplicação 67
3.20. Lista de Responsabilidades das Classes de Interfaces 68
3.21. Cartões das Classes de Interface 70
3.22. Lista de Responsabilidades das Classes de Persistência 71
3.23. Cartões das Classes de Persistência 72
3.24. Manual Preliminar do Usuário 72
3.25. Diagrama de Fluxo de Interfaces 84
3.26. Fase Exploratório com Cenários (Fec) 85
3.26.1. Responsabilidades por ordem de prioridade das Classes de Aplicação 85
3.26.2. Descrição das Responsabilidades e Colaborações 86
3.26.2. Relações de dependência das Responsabilidades entre as Classes de Aplicação 87
3.27. Diagrama de Classes (Fase de Consolidação e Relacionamento – Visão de Projeto) 87
3.28. Diagrama de Seqüência 89
3.29. Cartões Finais 91
3.30. Especificação da Estrutura de Dados das Classes de Aplicação 91
3.31. Especificação dos Métodos 93
3.32. Especificação da Implementação - Análise de Reuso 95
3.32.1. Importação de pacotes e reuso das classes de interface 96
3.33. Adaptação do Modelo Abstrato 101
3.34. Diagrama de Classes Adaptado – Visão de Projeto) 102
3.35. Diagrama de Execução 103
3.36. Codificação e Testes (Aplicação dos Padrões de Transição) 104
3.37. Codificação Detalhada 107
3.38. Plano de Testes 107
3.39. Análise de Risco 109
Conclusão 109
Anexos 113
5.1. Apostilas

5.2. Resultado do Workshop – Sistemas de Informação da EFEI (14 e 15/05/01)

6. Referências Bibliográficas 113

7. Bibliografia Complementar 114

"RESUMO"

Esta dissertação de Mestrado tem por finalidade, a apresentação de uma metodologia para o desenvolvimento de software do Sistema de informações administrativas da EFEI, concentrando a análise nas informações relacionadas ao seu Corpo Docente. A dissertação aborda técnicas de coletas de informações, o desenvolvimento de um modelo Conceitual (Modelo de Dados com base em técnicas de Modelagem de Dados Relacionais), a adaptação do modelo conceitual para o Modelo Relacional Normalizado, aplicações das Formas Normais, geração do modelo físico para implementação da Base de Dados em um Sistema Gerenciador de Banco de Dados – SGDB, documentação do software em UML enfocando a visão de caso de uso com o diagrama de caso de uso e visão de projeto com os diagramas de classes (para a modelagem estrutural) e de interação (para a modelagem comportamental).

Foram aplicados Métodos, Ferramentas e Procedimentos em Engenharia de Software com base no paradigma do Ciclo de Vida de Projeto Estruturado e Prototipação.

Este Projeto destina-se a atender parte do backlog (necessidades de desenvolvimento de Sistemas Computacionais) da Assessoria de Planejamento da Escola Federal de Engenharia de Itajubá.

"ABSTRACT"

The main purpose of this work is to present a new methodology for Software Development of the Information System at EFEI (Escola Federal de Engenharia de Itajubá), focusing on information related with its faculties. The dissertation approaches sampling data techniques, the development of a Conceptual Model (here named Data Model based on Relational Data Modelling), the adaptation of the Conceptual Model to a new Normalized Relational Model, application of Normal Forms, generation of the physical model to implement the database in a Database Management System – DBMS, the software documentation in UML with emphasis on the use-case view through the use-case diagram, and project view through the class diagram (for structural modelling) and the interaction diagram (for behaviour modelling).

Many Software Engineering Methods, Tools, and Procedures based on the life-cycle paradigm of Structured Project and Prototype were used in this work.

This work also intends to supply part of the backlog (demand for the development of computacional systems) required by the Planning Department of EFEI.

CAPÍTULO 1

1. INTRODUÇÃO

Qual o problema?

Hoje uma das maiores dificuldades enfrentada pela equipe técnica de desenvolvimento de software, é definir o melhor método, os procedimentos e as ferramentas a serem aplicadas no desenvolvimento de Software. Não obstante, a EFEI, até o momento, não dispõe de uma administração automatizada, dependendo de **procedimentos administrativos manuais** e de alguns **sistemas de informação** com características **não integradas**.

Tais sistemas não eliminam as **rotinas manuais** conduzindo à superposição, **duplicação de tarefas e de dados**, comprometendo o processo decisório devido à manipulação de dados inconsistentes.

Assim sendo, a não adoção **de uma metodologia de desenvolvimento de software para EFEI,** conduz a uma situação de imaturidade dos seus processos.

Qual a importância do problema?

Na página : http://www.mec.gov.br/sesu/ofertas.shtm pode-se avaliar a necessidade de todas as IFES se organizarem para atender as exigências legais que hoje se impõem. Dentre os itens avaliados em cada instituição a qualificação e o desempenho do CORPO DOCENTE com certeza ocupam situação de destaque.

"A Avaliação das Condições de Oferta de Cursos de Graduação é uma ação da Secretaria de Educação Superior (SESu) que visa a avaliar, de acordo com o disposto na Lei nº 9.131, de 24 de novembro de 1995, Decreto nº 2.026, de 10 de outubro de 1996, e Lei nº 9.394, de 20 de dezembro de 1996, de Diretrizes e Bases da Educação Nacional, *in loco*, cada um dos cursos de graduação submetidos ao Exame Nacional de Cursos (Provão), com relação à **qualificação de seu corpo docente**, à sua organização didático-pedagógica e às suas instalações, tanto as físicas em geral, quanto as especiais, tais como laboratórios, equipamentos e bibliotecas.

A avaliação periódica dos cursos e instituições de ensino superior, como determina a legislação, deve utilizar-se de procedimentos e critérios abrangentes com relação aos diversos fatores que determinam a qualidade e a eficiência das atividades de ensino, pesquisa e extensão."

Em nossa instituição os dados sobre o corpo docente são dispersos pelos diversos departamentos, não são atualizados com a freqüência necessária e são apresentados em diferentes modelos e configurações. Um sistema de informação unificado, eficiente, eficaz e atualizado, facilitaria a vida de todos departamentos que prestam informações além de possibilitar o acompanhamento, controle e avaliação constante da ação docente no ensino de graduação e pós-graduação, na pesquisa e na extensão.

Qual a proposta?

Desenvolver uma metodologia/padrão para o desenvolvimento de Software que permita visualização, especificação, construção e documentação de software's a serem desenvolvidos pela EFEI.

Segundo [Paula2000] a produção industrial de software é quase sempre uma atividade coletiva. Alguns produtos são construídos inicialmente por indivíduos ou por pequenas equipes. Na medida em que se tornam sucesso de mercado, passam a evoluir. A partir daí, um número cada vez maior de pessoas passa a cuidar da sua manutenção e evolução. Por isso, quase todas as atividades de Engenharia de Software são empreendidas por organizações.

[Gamma2000] afirma que todas as arquiteturas orientadas a objetos bem-estruturadas estão cheias de padrões. De fato, uma das maneiras de medir a qualidade de um sistema orientado a objetos é avaliar se os desenvolvedores tomaram bastante cuidado com as colaborações comuns entre seus objetos. Focalizar-se em tais mecanismos durante o desenvolvimento de um sistema pode levar a uma arquitetura menor, mais simples e muito mais compreensível que aquelas produzidas quando estes padrões são ignorados.

Este trabalho traz duas importantes contribuições. Primeiro, mostra o papel que a adoção de um metodologia no desenvolvimento de Sistemas pode exercer na arquitetura de sistemas complexos. Segundo, fornece uma referência muito prática da seqüência de Análise e documentação que o desenvolvedor atuante pode aplicar na criação de seus próprios projetos de Sistemas de Informação. Na implementação das contribuições acima foi usado no desenvolvimento, o Sistema de Informação dos Docentes das EFEI.

O que o leitor verá?

Este trabalho é dividido em 5 capítulos. O primeiro capítulo procura expressar a importância da Instituição de adotar uma metodologia de desenvolvimento de software. No segundo capítulo são apresentados as tecnologias aplicadas neste trabalho, como o modelo Entidade Relacionamento e UML(Unified Modeling Language). O terceiro capítulo é o desenvolvimento propriamente dito, com todos os passos propostos para o modelo, resultado da análise do estudo de caso: Sistema de informação do Corpo Docente, com aplicação das tecnologias ER e UML na prática. O quarto capítulo trata da conclusão do trabalho, o quinto os anexos e as referências bibliográficas.

Qual a aplicação feita?

Neste trabalho foi desenvolvido uma aplicação de Software para o Sistema de informações administrativas da EFEI em relação ao seu Corpo Docente, utilizando-se de técnicas de coletas de informações, bem como o desenvolvimento de um modelo Conceitual – Modelo de Dados com base em técnicas de Modelagem de Dados Relacionais; a adaptação do modelo conceitual para o Modelo Relacional Normalizado; aplicações das Formas Normais e desenvolvimento do modelo UML. Serão aplicados Métodos, Ferramentas e Procedimentos (Engenharia de Software), com base no paradigma do Ciclo de Vida de Projeto Estruturado.

Este Projeto destina-se a atender parte do backlog (Necessidades de desenvolvimento de Sistemas Computacionais) da Assessoria de Planejamento da Escola Federal de Engenharia de Itajubá.

CAPÍTULO 2

2. TECNOLOGIAS APLICADAS AO PROJETO

TÉCNICAS DE COLETA DE INFORMAÇÕES

Durante a fase de desenvolvimento do sistema, vários tipos de usuários serão entrevistados. As principais razões são:

- Coletar informações sobre o comportamento do sistema atual e/ou os requisitos para o novo sistema;
- Verificar a nossa própria compreensão sobre o sistema atual e/ou os requisitos para o novo sistema;
- Coletar informações sobre o sistema atual para estudar Custo x Benefício .

As reuniões podem ser necessárias para:

- Informar
- Gerar idéias
- Definir metas
- Avaliar
- Resolver problemas
- Planejar ou definir estratégias
- Tomar decisões
- Obter comprometimento
- Obter resultados
- **Obter requisitos de sistemas**

[Costa1994] cita, na década de 80 popularizou-se uma técnica denominada JAD (Joint Application Development - Desenvolvimento Conjunto de Aplicação) ou Técnica de Projeto Acelerado. Consiste de reuniões rápidas (e Intensivas) num processo acelerado de coleta de dados. Analistas e usuários participam destas reuniões com o objetivo de delimitar os requisitos do sistema. São supervisionadas por um especialista chamado **mediador**.

[Pressman1995] **Demarco** afirma que;

"Análise de Sistemas é uma forma de comunicação entre estranhos". Usuários e analistas têm vocabulários e experiências diferentes, e muitas vezes, um diferente conjunto de pressuposições, percepções, valores e prioridades.

DIRETRIZES PARA A REALIZAÇÃO DE ENTREVISTAS

a) ***Desenvolva um Plano Geral de Entrevistas***

b) ***Certifique-se de que você tem autorização para falar com os usuários***

c) ***Planeje a entrevista para fazer uso eficiente do tempo***

d) ***Utilize ferramentas automatizadas que sejam adequadas, mas não abuse***

e) ***Tente descobrir em que informação o usuário está mais interessado***

PROJETANDO A REUNIÃO DE QUE VOCÊ PRECISA

Projetar cada tipo específico de reunião, de modo que a mesma tenha um clima apropriado assegurando a produtividade.

Exemplos de tipos de reuniões:

- Informativa (disseminar informações);
- Motivação (motivar o pessoal);
- Brainstorming (geração de idéias);
- JAD (obter requisitos).

ACOMPANHANDO UMA REUNIÃO

Podemos identificar 4 (quatro) estágios de uma reunião:

- Planejamento
- Início
- Estrutura e Controle
- Encerramento

PLANEJAMENTO

Para fazer o planejamento da reunião, a primeira pergunta a ser feita é:

"Precisamos de uma reunião para atingir este objetivo?"

Se realmente for necessária, precisamos levar em consideração:

a) Reservar o local mais adequado para a reunião

b) Planejar e certificar-se que os recursos audiovisuais estarão disponíveis

c) Publicar antecipadamente a agenda da reunião, certificando-se que todos os participantes foram informados.

A agenda é importante para assegurar que cada participante compareça à reunião preparado para esclarecer as expectativas e o papel dos participantes e para estruturar: conteúdo, processo e tempo de reunião.

INÍCIO

Observar os seguintes pontos:

a) Começar no horário

b) Apresentar a agenda

c) Obter o envolvimento compromisso dos participantes através da revisão da agenda.

ESTRUTURA E CONTROLE

Os erros mais comuns são:

- A reunião não segue a agenda;
- O líder passa de um tópico a outro de forma confusa;
- Os participantes fazem perguntas conducentes, criticantes ou múltiplas;
- A discussão vagueia longe do assunto;
- O líder força o acordo;
- Os problemas de processo não são abordados na reunião;
- Idéias são perdidas porque não são registradas.

Para evitar que as idéias e resoluções tomadas na reunião fiquem esquecidas é necessário elaborar durante a reunião uma memória do grupo. Esta memória serve para:

- Tornar pública as minutas da reunião;
- Registar as idéias de cada participante;
- Tornar visíveis as decisões, as ações e acompanhamento da reunião.

ENCERRAMENTO

Os erros mais comuns quanto ao encerramento da reunião são:

- Falta de encerramento, causando frustrações;
- Atribuições são confusas ou não são feitas;
- O líder assume que os participantes têm um entendimento comum que ocorreu na reunião;

- Ações necessárias não são consideradas.

Métodos que podem ser utilizados no encerramento da reunião:

MÉTODO	BENEFÍCIO
Perguntas – resumo: - A que conclusões podemos chegar? - Quais são os próximos passos? - Quem irá fazer o quê e Quando?	Estabelece planos de ação. Define responsabilidades individuais.
Perguntas – fechadas para determinar se a reunião atingiu seus objetivos: - Completamos nossa tarefa? - Todos estamos satisfeitos com os resultados desta reunião?	Evita erros de interpretações ao terminar a reunião muito rapidamente.
Registar na memória do grupo quais decisões, itens de ação, responsabilidades e prazos.	Assegurar que não haverá surpresas no resumo da reunião. Dar ao grupo o senso de realização.
Avaliar a reunião.	Dar ao líder um feedback útil de como foi a reunião. Dar aos participantes uma oportunidade de comentar anonimamente a reunião.
Enviar uma memória do grupo para cada participante no devido tempo.	Lembrar aos participantes suas responsabilidades.

INTRODUÇÃO – SISTEMA: UMA VISÃO GERAL

Sistema → grupo de itens que interagem entre si ou que sejam interdependentes, formando um todo unificado. Ordem, Organização e harmonia.

Sistemas automatizados são aqueles que interagem com ou são controlados pôr um ou mais computadores.

COMPONENTES dos Sistemas Automatizados:

_ Hardware: terminais, impressoras, CPD, ...

_ Software: sistemas operacionais, sistemas de banco de dados, ...

_ Peopleware (humanware, Brainware): pessoas envolvidas no processo de desenvolvimento e operação dos sistemas;

_ Dados: informações que o sistema mantém pôr um período de tempo;

_ Procedimentos: serviços, instruções formais para a operação do sistema.

MODELAGEM LÓGICA DE SISTEMAS

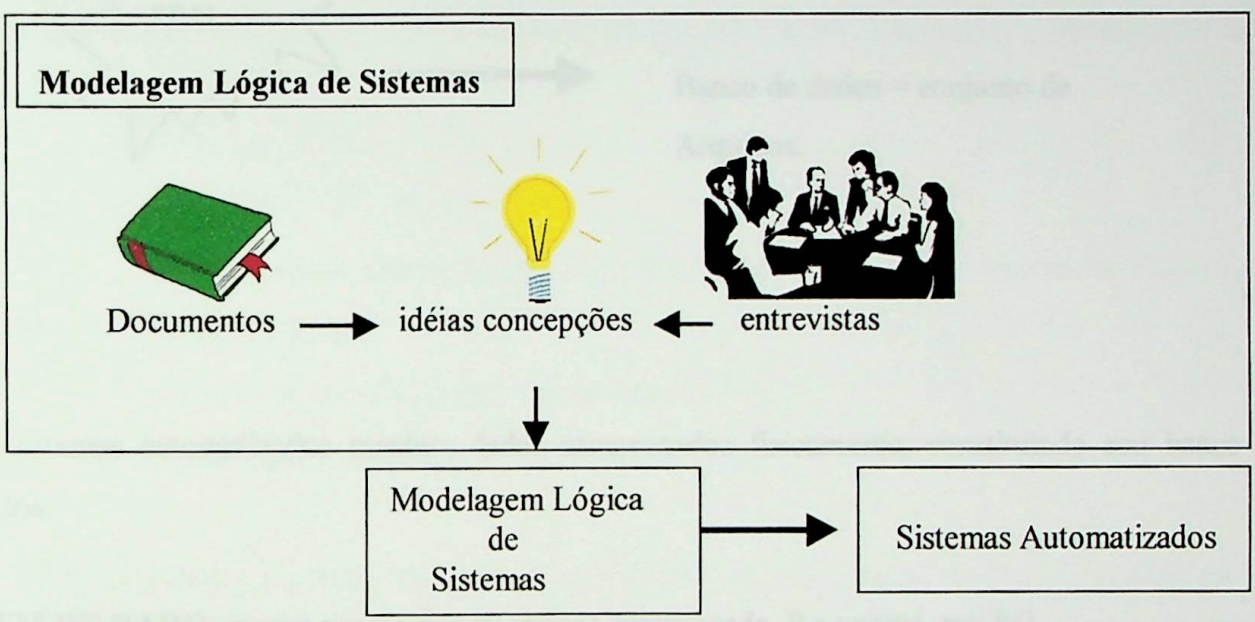
O que é um MODELO?

Modelo é uma representação abstrata do que se tornará uma combinação de hardware e software.

Um modelo RECRIA a realidade observada através dos sentidos.

Assim, MODELAGEM refere-se à técnica utilizada na criação de modelos.

A Modelagem Lógica não leva em consideração aspectos físicos do sistema, tais como: especificação de equipamentos, de softwares, ...



O Objetivo da Modelagem Lógica de Sistemas é receber idéias vagas sobre necessidades e transformá-las, o mais rápido possível, em definições precisas[Chen1990].

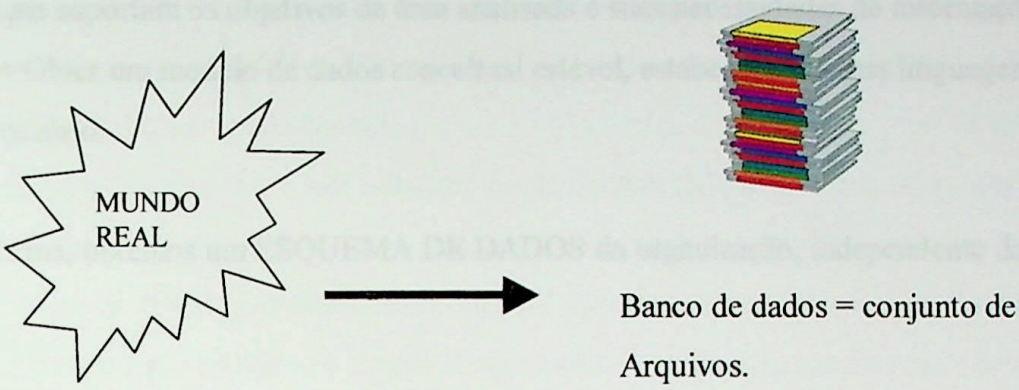
Modelagem de Dados → É a representação das informações necessárias para um Sistema Automatizado em relação a um determinado contexto.

Atividades Relativas à Modelagem de Dados:

- Definição de Dados
- Definição de Relacionamentos entre os Dados
- Normalização
- Construção do Diagrama Entidade-Relacionamento

BANCO DE DADOS RELACIONAL

[Rumbaugh1994] O paradigma baseado em objetos é versátil. Ele não só obedece a uma sólida base para o projeto de sistemas e para a programação como também pode ser usado para projetar bancos de dados. A utilização de um projeto baseado em objetos transcende a escolha do banco de dados.



Os sistemas automatizados mantém dados armazenados fisicamente, constituindo seu banco de Dados.

ITEM DE DADO: menor porção de informação armazenada. P.e.: nome, rua RG.
REGISTRO: coleção de itens de dados. Ex: registro de funcionário: nome, RG e sexo.
ARQUIVO: coleção de registros do mesmo tipo:
BANCO DE DADOS: coleção de arquivos, que podem estar ou não interrelacionados.

SISTEMAS DE BANCO DE DADOS E MODELOS DE DADOS

SISTEMA DE GERÊNCIA DE BANCO DE DADOS – SGDB

Segundo [Rumbaugh1994] um Sistema de Gerenciamento de Banco de Dados é um conjunto de programas utilizados para a *construção, utilização e manutenção* de um Banco de Dados.

Problemas Associados ao Projeto de Bancos de Dados:

1. O projetista de BD é restringido pelas estruturas oferecidas pelo Sistema de Banco de Dados;
2. O projetista deve considerar aspectos de performance para o BD, tanto em recuperação quanto em atualização do mesmo.

O esquema produzido não leva em consideração aspectos físicos.

MODELAGEM E SEUS OBJETIVOS

Os objetivos da Modelagem de Dados são:

- Expandir e detalhar um modelo de dados parcial (informação), identificando todos os dados que suportam os objetivos da área analisada e suas necessidades de informações;
- Obter um modelo de dados conceitual estável, estabelecendo uma linguagem comum com o usuário.

Desta forma, obtemos um ESQUEMA DE DADOS da organização, independente da implementação física.

Vantagens:

- O modelo de dados não contém restrições características do Sistema de Gerência de Banco de Dados;
- Esquema da organização é mais estável;
- Representação gráfica de fácil compreensão.

Abstrair: concentrar a atenção em pontos que realmente interessam, encapsulando detalhes em representações, para que possam ser utilizadas posteriormente.

Assim, a modelagem de dados constitui-se numa forma de abstração.

UML (UNIFIED MODELING LANGUAGE)

A Unified Modeling Language (UML) é resultado do esforço conjunto das metodologias Grady Booch, James Rumbaugh e Ivar Jacobson, e de um consórcio de empresas interessadas em modelagem orientada a objetos. A partir de dezembro/1997, a UML foi adotada como linguagem de modelagem padrão pelo Object Management Group (OMG).

INTRODUÇÃO

Segundo [Booch1997] o grande problema do desenvolvimento de novos sistemas utilizando a orientação a objetos nas fases de análise de requisitos, análise de sistemas e design é que não existe uma notação padronizada e realmente eficaz que abranja qualquer tipo de aplicação que se deseje.

[Rumbaugh1994] afirma que a saída da análise é um modelo formal que incorpora os três aspectos essenciais do sistema: os objetos e o relacionamento entre eles, o fluxo dinâmico de controle e a transformação funcional dos dados sujeita a restrições.

A UML é muito mais que a padronização de uma notação. É também o desenvolvimento de novos conceitos não usados normalmente. Por isso e muitas outras razões, o bom entendimento da UML não é apenas aprender a simbologia e o seu significado, mas também significa aprender a modelar orientado a objetos no estado da arte.

UML foi desenvolvida por Grady Booch, James Rumbaugh, e Ivar Jacobson que são conhecidos como "os três amigos". Eles possuem um extenso conhecimento na área de modelagem orientado a objetos já que as três mais conceituadas metodologias de modelagem orientado a objetos foram eles que desenvolveram e a UML é a junção do que havia de melhor nestas três metodologias adicionado novos conceitos e visões da linguagem.

UML – A UNIFICAÇÃO DOS MÉTODOS PARA A CRIAÇÃO DE UM NOVO PADRÃO

A UML é uma tentativa de padronizar a modelagem orientada a objetos de uma forma que qualquer sistema, seja qual for o tipo, possa ser modelado corretamente, com consistência, fácil de se comunicar com outras aplicações, simples de ser atualizado e compreensível.

Existem várias metodologias de modelagem orientada a objetos que até o surgimento da UML causavam uma guerra da comunidade de desenvolvedores orientado a objetos. A UML acabou com esta guerra trazendo as melhores idéias de cada uma destas metodologias, e mostrando como deveria ser a migração de cada uma para a UML.

USO DA UML

A UML é usada no desenvolvimento dos mais diversos tipos de sistemas. Ela abrange sempre qualquer característica de um sistema em um de seus diagramas e é também aplicada em diferentes fases do desenvolvimento de um sistema, desde a especificação da análise de requisitos até a finalização com a fase de testes.

O objetivo da UML é descrever qualquer tipo de sistema, em termos de diagramas orientado a objetos. Naturalmente, o uso mais comum é para criar modelos de sistemas de software, mas a UML também é usada para representar sistemas mecânicos sem nenhum software.

FASES DO DESENVOLVIMENTO DE UM SISTEMA EM UML

Existem cinco fluxos de trabalho (etapas) no desenvolvimento de sistemas de software: análise de requisitos, análise, design (projeto), programação e testes. Estas cinco etapas não devem ser

executadas na ordem descrita acima, mas concomitantemente de forma que problemas detectados numa certa fase modifiquem e melhorem as fases desenvolvidas anteriormente de forma que o resultado global gere um produto de alta qualidade e performance [Eriksson1998]. A seguir falaremos sobre cada fase do desenvolvimento de um sistema em UML:

ANÁLISE DE REQUISITOS

Esta etapa captura as intenções e necessidades dos usuários do sistema a ser desenvolvido através do uso de funções chamadas "use-cases". Através do desenvolvimento de "use-case", as entidades externas ao sistema (em UML chamados de "atores externos") que interagem e possuem interesse no sistema são modelados entre as funções que eles requerem, funções estas chamadas de "use-cases".

ANÁLISE

A etapa de análise está preocupada com as primeiras abstrações (classes e objetos) e mecanismos que estarão presentes no domínio do problema. As classes são modeladas e ligadas através de relacionamentos com outras classes, e são descritas no Diagrama de Classe. As colaborações entre classes também são mostradas neste diagrama para desenvolver os "use-cases" modelados anteriormente, estas colaborações são criadas através de modelos dinâmicos em UML.

DESIGN (PROJETO)

Na etapa de design, o resultado da análise é expandido em soluções técnicas. Novas classes serão adicionadas para prover uma infra-estrutura técnica: a interface do usuário e de periféricos, gerenciamento de banco de dados, comunicação com outros sistemas, dentre outros.

PROGRAMAÇÃO

Na etapa de programação, as classes provenientes do design são convertidas para o código da linguagem orientada a objetos escolhida (a utilização de linguagens procedurais é extremamente não recomendada). Dependendo da capacidade da linguagem usada, essa conversão pode ser uma tarefa fácil ou muito complicada.

TESTES

Um sistema normalmente é rodado em testes de unidade, integração, e aceitação. Os testes de unidade são para classes individuais ou grupos de classes e são geralmente testados pelo programador. Os testes de integração são aplicados já usando as classes e componentes integrados para se confirmar se as classes estão cooperando uma com as outras como especificado nos modelos.

A NOTAÇÃO DA LINGUAGEM DE MODELAGEM UNIFICADA – UML

Tendo em mente as quatro fases do desenvolvimento de softwares, as fases de concepção, elaboração, construção e transição utilizam-se em seu desenvolvimento cinco tipos de visões, nove tipos de diagramas e vários modelos de elementos que serão utilizados na criação dos diagramas e mecanismos gerais que, todos em conjunto, especificam e exemplificam a definição do sistema, tanto a definição no que diz respeito à funcionalidade estática, como a dinâmica do desenvolvimento de um sistema.

Antes de abordarmos cada um destes componentes separadamente, definiremos as partes que compõem a UML:

- Visões;
- Modelos de Elementos;
- Mecanismos Gerais;
- Diagramas.

VISÕES

Um único gráfico não é capaz de capturar todas as informações necessárias para descrever um sistema.

Um sistema é composto por diversos aspectos: *funcional* (que é sua estrutura estática e suas interações dinâmicas), *não funcional* (requisitos de tempo, confiabilidade, desenvolvimento, etc.) e aspectos organizacionais (organização do trabalho, mapeamento dos módulos de código, etc.).

Cada visão é descrita por um número de diagramas que contém informações que dão ênfase aos aspectos particulares do sistema. Os diagramas que compõem as visões contém os modelos de elementos do sistema. As visões que compõem um sistema são:

- Visão "use-case": Descreve a funcionalidade do sistema desempenhada pelos atores externos do sistema (usuários).
- Visão Lógica: Descreve como a funcionalidade do sistema será implementada. É feita principalmente pelos analistas e desenvolvedores.
- Visão de Componentes: É uma descrição da implementação dos módulos e suas dependências. É principalmente executada por desenvolvedores, e consiste nos componentes dos Sistemas.
- Visão de concorrência: Trata a divisão do sistema em processos e processadores. Este aspecto, que é uma propriedade não funcional do sistema, permite uma melhor utilização do

ambiente onde o sistema se encontrará, se o mesmo possui execuções paralelas, e se existe dentro do sistema um gerenciamento de eventos assíncronos.

- Visão de Organização: Finalmente, a visão de organização mostra a organização física do sistema, os computadores, os periféricos e como eles se conectam entre si. Esta visão será executada pelos desenvolvedores, integradores e testadores, e será representada pelo diagrama de execução.

MODELOS DE ELEMENTOS

Os conceitos usados nos diagramas são chamados de modelos de elementos. Um modelo de elemento é definido com a semântica, a definição formal do elemento com o exato significado do que ele representa, sem definições duvidosas ou ambíguas e também define sua representação gráfica que é mostrada nos diagramas da UML.

CLASSES

Uma classe é a descrição de um tipo de objeto. Todos os objetos são instâncias de classes, onde a classe descreve as propriedades e comportamentos daquele objeto. Objetos só podem ser instanciados de classes.

OBJETOS

Um objeto é um elemento que podemos manipular, acompanhar seu comportamento, criar, destruir etc. Um objeto existe no mundo real.

ESTADOS

Todos os objetos possuem um estado que significa o resultado de atividades executadas pelo objeto, e é normalmente determinado pelos valores de seus atributos e ligações com outros objetos.

PACOTES

Pacote é um mecanismo de agrupamento, onde todos os modelos de elementos podem ser agrupados.

COMPONENTES

Um componente pode ser tanto um código em linguagem de programação como um código executável já compilado. Por exemplo, em um sistema desenvolvido em Java, cada arquivo *.java* ou *.class* é um componente do sistema, e será mostrado no diagrama de componentes que os utiliza.

RELACIONAMENTOS

Os relacionamentos ligam as classes/objetos entre si criando relações lógicas entre estas entidades. Os relacionamentos podem ser dos seguintes tipos:

- Associação;
- Generalização;
- Dependência e Refinamentos.

ASSOCIAÇÕES

Uma associação representa que duas classes possuem uma ligação (link) entre elas, significando por exemplo que elas "conhecem uma à outra", "estão conectadas com", "para cada X existe um Y" e assim por diante.

ASSOCIAÇÕES NORMAIS

O tipo mais comum de associação é apenas uma conexão entre classes. É representada por uma linha sólida entre duas classes. A associação possui um nome (junto à linha que representa a associação), normalmente um verbo, mas substantivos também são permitidos.

ASSOCIAÇÃO RECURSIVA

É possível conectar uma classe a ela mesma através de uma associação e que ainda representa semanticamente a conexão entre dois objetos, mas os objetos conectados são da mesma classe.

ASSOCIAÇÃO QUALIFICADA

Associações qualificadas são usadas com associações de um para vários (1..*) ou vários para vários (*). O "qualificador" (identificador da associação qualificada) especifica como um determinado objeto no final da associação "n" é identificado, e pode ser visto como um tipo de chave para separar todos os objetos na associação.

ASSOCIAÇÃO EXCLUSIVA

Em alguns modelos nem todas as combinações são válidas, e isto pode causar problemas que devem ser tratados. Uma associação exclusiva é uma restrição em duas ou mais associações.

ASSOCIAÇÃO ORDENADA

As associações entre objetos podem ter uma ordem implícita. O padrão para uma associação é desordenada (ou sem nenhuma ordem específica). Mas uma ordem pode ser especificada através da associação ordenada.

ASSOCIAÇÃO DE CLASSE

Uma classe pode ser associada a uma outra associação. Este tipo de associação não é conectada a nenhuma das extremidades da associação já existente, mas na própria linha da associação. Esta associação serve para se adicionar informações extras a associação já existente.

ASSOCIAÇÃO TERNÁRIA

Mais de duas classes podem ser associadas entre si, a associação ternária associa três classes. Ela é mostrada como um grade losango (diamante) e ainda suporta uma associação de classe ligada a ela, traçaria-se, então, uma linha tracejada a partir do losango para a classe onde seria feita a associação ternária.

AGREGAÇÃO

A agregação é um caso particular da associação. A agregação indica que uma das classes do relacionamento é uma parte, ou está contida em outra classe. As palavras-chave usadas para identificar uma agregação são: "consiste em", "contém", "é parte de".

GENERALIZAÇÕES

A generalização é um relacionamento entre um elemento geral e um outro mais específico. O elemento mais específico possui todas as características do elemento geral e contém ainda mais particularidades. Um objeto mais específico pode ser usado como uma instância do elemento mais geral. A generalização, também chamada de herança, permite a criação de elementos especializados em outros.

DEPENDÊNCIAS E REFINAMENTOS

Além das associações e generalizações, existem ainda dois tipos de relacionamentos em UML. O relacionamento de dependência é uma conexão semântica entre dois modelos de elementos, um independente e outro dependente. Uma mudança no elemento independente irá afetar o modelo dependente.

MECANISMOS GERAIS

A UML utiliza alguns mecanismos em seus diagramas para tratar informações adicionais.

- Ornamentos;
- Notas.

DIAGRAMAS

Os diagramas utilizados pela UML são compostos de nove tipos: diagrama de use case, de classes, de objeto, de estado, de seqüência, de colaboração, de atividade, de componente e o de execução. Considerando a modelagem de um sistema distribuído e complexo, será necessário empregar todos os tipos de diagramas da UML para expressar a arquitetura do seu sistema e os riscos técnicos do projeto, conforme é apresentado adiante.

· Visão de caso de uso	Diagramas de caso de uso (para a modelagem comportamental)
	Diagramas de atividades (para a modelagem comportamental)
· Visão de projeto	Diagramas de classes (para a modelagem estrutural)
	Diagramas de colaboração (para a modelagem comportamental)
	Diagramas de estados (para a modelagem comportamental)
· Visão de processo	Diagramas de objeto (para a modelagem estrutural)
	Diagramas de seqüência (para a modelagem comportamental)
· Visão de implementação	Diagramas de componentes (para a modelagem estrutural)
· Visão de implantação	Diagramas de execução (para a modelagem estrutural)

DIAGRAMAS ESTRUTURAIS

Os quatro diagramas estruturais da UML existem para visualizar, especificar, construir e documentar os **aspectos estáticos** de um sistema [Booch 2000]. Considere os aspectos estáticos do sistema como uma representação de seu esqueleto e estrutura relativamente estáveis. Assim como os aspectos estáticos de uma casa incluem a existência e a colocação de itens como paredes, portas, janelas, canos, fios e respiradouros, também os aspectos estáticos de um sistema de software abrangem a existência e a colocação de itens como classes, interfaces, colaborações, componentes e nós.

1. Diagrama de classes
2. Diagrama de objetos
3. Diagrama de componentes
4. Diagrama de implantação/execução

DIAGRAMAS COMPORTAMENTAIS

Os cinco diagramas comportamentais da UML são utilizados para visualizar, especificar, construir e documentar os **aspectos dinâmicos** de um sistema[Booch2000].

Com freqüência, você usará cinco diagramas adicionais para visualizar as partes dinâmicas de um sistema.

1. Diagrama de caso de uso
2. Diagrama de seqüências
3. Diagrama de colaboração
4. Diagrama de gráfico de estados
5. Diagrama de atividades

DIAGRAMA USE-CASE

A modelagem de um diagrama use-case é uma técnica usada para descrever e definir os requisitos funcionais de um sistema. Eles são escritos em termos de atores externos, use-cases e o sistema modelado. Os atores representam o papel de uma entidade externa ao sistema como um usuário, um hardware, ou outro sistema que interage com o sistema modelado.

Um diagrama de caso de uso mostra um conjunto de casos de uso e atores (um tipo especial de classe) e seus relacionamento.

DIAGRAMA DE CLASSES

O diagrama de classes demonstra a estrutura estática das classes de um sistema onde estas representam as "coisas" que são gerenciadas pela aplicação modelada. Classes podem se relacionar com outras através de diversas maneiras: associação (conectadas entre si), dependência (uma classe depende ou usa outra classe), especialização (uma classe é uma especialização de outra classe), ou em pacotes (classes agrupadas por características similares).

Um diagrama de classes mostra um conjunto de classes, interfaces e colaborações e seus relacionamentos.

DIAGRAMA DE OBJETOS

O diagrama de objetos é uma variação do diagrama de classes e utiliza quase a mesma notação. A diferença é que o diagrama de objetos mostra os objetos que foram instanciados das classes. O diagrama de objetos é como se fosse o perfil do sistema em um certo momento de sua execução.
Um diagrama de objetos mostra um conjuntos de objetos e seus relacionamentos. Use esses diagramas para ilustrar as estruturas de dados, registros estáticos de instâncias dos itens encontrados nos diagramas classes.

DIAGRAMA DE ESTADO

O diagrama de estado é tipicamente um complemento para a descrição das classes. Este diagrama mostra todos os estados possíveis em que objetos de uma certa classe podem se encontrar e mostra também quais são os eventos do sistemas que provocam tais mudanças.
Um diagrama de gráfico de estados mostra uma máquina de estados, que consiste de estados, transições, eventos e atividades. Use os diagramas de gráfico de estados para ilustrar a visão dinâmica de um sistema. Esses diagramas são importantes principalmente para se fazer a modelagem do comportamento de uma interface, classe ou colaboração.

DIAGRAMA DE SEQUÊNCIA

Um diagrama de seqüência mostra a colaboração dinâmica entre os vários objetos de um sistema. O mais importante aspecto deste diagrama é que a partir dele percebe-se a seqüência de mensagens enviadas entre os objetos. Ele mostra a interação entre os objetos, alguma coisa que acontecerá em um ponto específico da execução do sistema.
Um diagrama de seqüência é um diagrama de interação que dá ênfase à ordenação temporal de mensagens. Um diagrama de seqüência mostra o conjunto de objetos e as mensagens enviadas e recebidas por esses objetos.

DIAGRAMA DE COLABORAÇÃO

Um diagrama de colaboração mostra de maneira semelhante ao diagrama de seqüência, a colaboração dinâmica entre os objetos. Normalmente pode-se escolher entre utilizar o diagrama de colaboração ou o diagrama de seqüência.
Um diagrama de colaboração é um diagrama de interação que dá ênfase à organização estrutural dos objetos que enviam e recebem mensagens. Um diagrama de colaboração mostra um conjunto de objetos, as conexões existentes entre esses objetos e as mensagens enviadas e recebidas pelos objetos.

DIAGRAMA DE ATIVIDADE

Diagramas de atividade capturam ações e seus resultados. Eles focam o trabalho executado na implementação de uma operação (método), e suas atividades numa instância de um objeto. O diagrama de atividade é uma variação do diagrama de estado e possui um propósito um pouco diferente do diagrama de estado, que é o de capturar ações (trabalho e atividades que serão executados) e seus resultados em termos das mudanças de estados dos objetos.

Um diagrama de atividades mostra um conjunto de atividades, o fluxo seqüencial ou ramificado de uma atividade para outra e os objetos que realizam ou sofrem ações.

DIAGRAMA DE COMPONENTE

O diagrama de componente e o de execução são diagramas que mostram o sistema por um lado funcional, expondo as relações entre seus componentes e a organização de seus módulos durante sua execução.

O diagrama de componente descreve os componentes de software e suas dependências entre si, representando a estrutura do código gerado.

Um diagrama de componentes mostra um conjunto de componentes e seus relacionamentos.

DIAGRAMA DE EXECUÇÃO

O diagrama de execução mostra a arquitetura física do hardware e do software no sistema. Pode mostrar os atuais computadores e periféricos, juntamente com as conexões que eles estabelecem entre si e pode mostrar também os tipos de conexões entre esses computadores e periféricos. Especificam-se também os componentes executáveis e objetos que são alocados para mostrar quais unidades de software são executados e em quais destes computadores são executados.

Um diagrama de implantação mostra conjunto de nós e seus relacionamentos.

CAPÍTULO 3

DESENVOLVIMENTO DO SOFTWARE

Segundo [Booch2000], existem quatro fases no ciclo de desenvolvimento de software: concepção, elaboração, construção e transição. Os fluxos são representados nessas fases, indicando as variações de graus do foco ao longo do tempo.

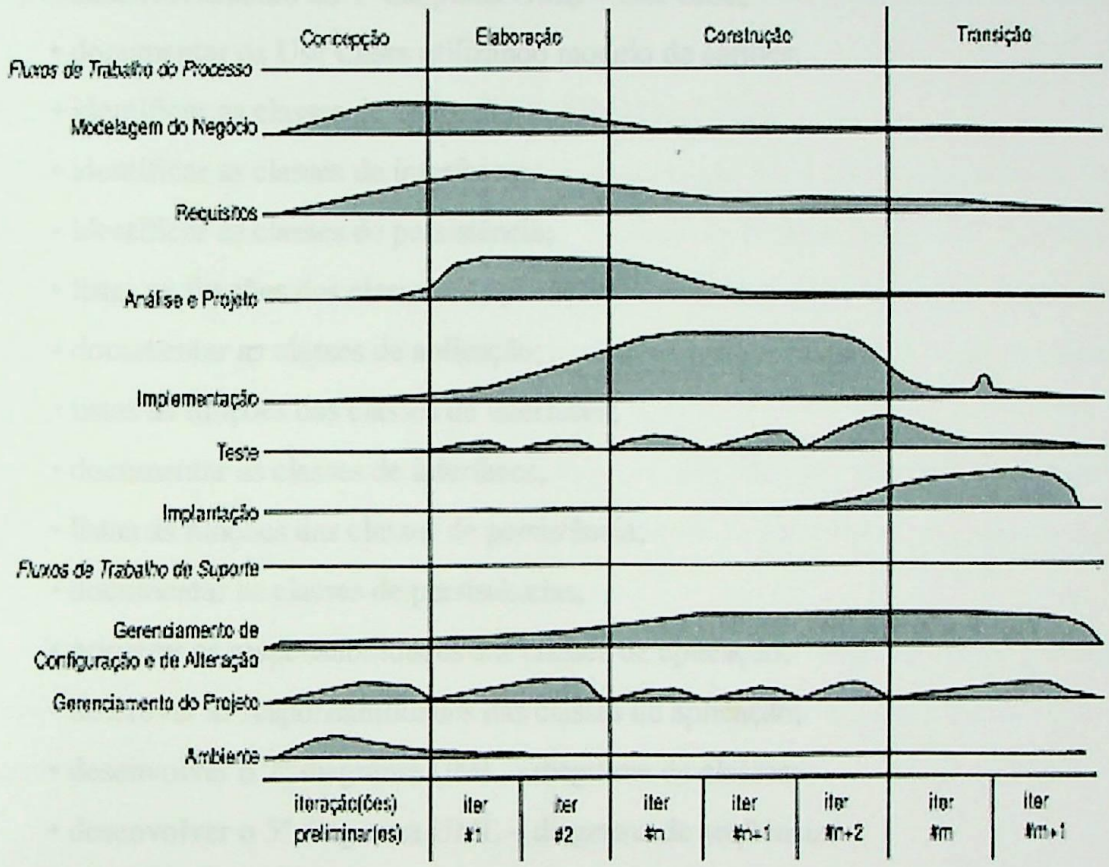


Figura 2.21: Ciclo de vida de desenvolvimento de um software

ESTRATÉGIA DO DESENVOLVIMENTO DO PROJETO

Seguiremos como estratégia de desenvolvimento deste trabalho as seguintes etapas associadas as fases do ciclo de desenvolvimento de software:

Fase de Concepção

- primeiramente a coleta de informações com a execução da fase de entrevistas;
- elaboração da proposta de desenvolvimento;

Fase de Elaboração

- desenvolvimento do modelo descritivo;
- identificar as entidades representando que devem fazer parte do sistema;
- modelagem do sistema – DER;
- modelo relacional normalizado – ErWin;
- modelo de dicionário de dados;
- identificação dos clientes/atores;
- identificação das funções;
- desenvolvimento do 1° diagrama UML – Use Case;
- documentar os Use Cases utilizando modelo de cartões;
- identificar as classes de aplicação;
- identificar as classes de interfaces;
- identificar as classes de persistência;
- listas as funções das classes de aplicação;
- documentar as classes de aplicação;
- listas as funções das classes de interfaces;
- documentar as classes de interfaces;
- listas as funções das classes de persistência;
- documentar as classes de persistências;
- priorizar as responsabilidades das classes de aplicação;
- descrever as responsabilidades das classes de aplicação;
- desenvolver o 2° diagrama UML – diagrama de classes;
- desenvolver o 3° diagrama UML – diagrama de seqüência;
- descrever últimos cartões;
- especificar estruturas de dados;
- especificar métodos;
- desenvolver diagrama de classes completo;
- desenvolver o 4° diagrama UML – diagrama de execução;

Fase de Construção

- apresentar a codificação;
- apresentar um plano de teste e análise de riscos, conclusão e estimativas futuras.

ESCOLHA DOS DIAGRAMAS

Na modelagem de sistemas reais, independentemente do domínio do problema, deverão ser criados os mesmos tipos de diagramas, pois eles representam visões de modelos comuns [Booch2000].

Considerando o grau de complexidade médio do Domínio do problema em estudo, desenvolveremos os seguintes diagramas:

· Visão de caso de uso	Diagramas de caso de uso(para a modelagem comportamental)
· Visão de projeto	Diagramas de classes (para a modelagem estrutural)
	Diagramas de seqüência (para a modelagem comportamental)
· Visão de implantação	Diagramas de execução(para a modelagem estrutural)

3.1. ESTUDO INICIAL – COLETA DE INFORMAÇÕES

Na década de 80 popularizou-se uma técnica denominada JAD (Joint Application Development - Desenvolvimento Conjunto de Aplicação) ou Técnica de Projeto Acelerado. Consiste de reuniões rápidas (e Intensivas) num processo acelerado de coleta de dados. Analistas e usuários participam destas reuniões com o objetivo de delimitar os requisitos do sistema. São supervisionadas por um especialista chamado **mediador**.

Nesta fase não iremos usar o modelo JAD propriamente dito, por motivos de limitação da equipe envolvida. Iremos executar seções de trabalho e através de entrevista, perguntas e respostas, serão extraídos os primeiros requisitos do Sistema.

Neste item serão documentadas todas as entrevistas feitas com os usuários identificados como envolvidos com o Sistema de Informação do Docente. Segue abaixo um modelo de formulário e as perguntas a serem feitas. Esta reunião não deve ultrapassar 1 hora.

MODELO DE FORMULÁRIO DE ENTREVISTA UTILIZADO

FORMULÁRIO DE ESPECIFICAÇÃO DE REQUISITOS	
Identificação: FL01 Analistas: Participantes: Obs:	Local: **Data:** **Hora:**

Perguntas:

1. Que atividades são executadas neste departamento?
2. Quem são as pessoas responsáveis por cada atividade?
3. Quem pode ser considerado o responsável por todas as atividades? (o Diretor)
4. Na possível montagem de um Sistema de armazenamento de informações (B.D.) neste departamento, quem usará a solução?
5. Qual(is) o(s) benefício(s) no seu ver de uma solução bem-sucedida ?
6. Você enxerga outra fonte para a solução de armazenamento de informações ?
7. Como você caracterizaria um "bom" resultado (saída) que seria gerado pôr uma solução bem-sucedida?
8. Que documentos você utiliza nas atividades: documentos, anotações, formulários, relatórios,
9. Você poderia mostrar-me (ou descrever-me) o ambiente em que a solução será usada?
10. Quais informações serão produzidas?
11. Quais informações devem ser fornecidas?
12. Quais informações devem ser armazenadas?
13. Que funções e desempenho são exigidos?
14. Existe tecnologia para construir o sistema?
15. Quais recursos especiais de desenvolvimento e produção serão exigidos?
16. Quais limites foram estabelecidos para os custos e para os prazos?
17. Qual é o mercado em potencial para o produto?
18. Como esse produto se compara com os produtos dos concorrentes?
19. Que posição esse produto ocupa na linha de produtos global da empresa?

Respostas:

FORMULÁRIO DE ESPECIFICAÇÃO DE REQUISITOS SISTEMA DE INFORMAÇÃO DO CORPO DOCENTE	
Identificação: FL01 Analista: Cláudio Neves Sá Participantes: Catarina / PDG Obs:	Local: CPD **Data: 20/11/2000** **Hora: 09:00h**
Identificação: FL02 Analista: Cláudio Neves Sá Participantes: Luiz Cláudio / ASP Obs:	Local: ASP **Data: 20/11/2000** **Hora: 15:00h**
Identificação: FL03 Analista: Cláudio Neves Sá Participantes: Maria Helena / DEP Obs:	Local: DEP **Data: 21/11/2000** **Hora: 09:30h**
Identificação: FL04 Analista: Cláudio Neves Sá Participantes: Prof. Paulo Shiguemi /IEM Obs:	Local: IEM **Data: 21/11/2000** **Hora: 14:00h**
Identificação: FL05 Analista: Cláudio Neves Sá Participantes: Prof. Germano /IEE Obs:	Local: PPG **Data: 22/11/2000** **Hora: 09:00h**

Perguntas:

1. Que atividades são executadas neste departamento em relação a informação do corpo docente?
2. Quem são as pessoas responsáveis por cada atividade?
3. Você pode descrever-me cada atividade citada acima?
4. Na possível montagem de um Sistema de armazenamento de informações (B.D.) neste departamento, quem usará a solução?
5. Qual(is) o(s) benefício(s) no seu ver de uma solução bem-sucedida ?
6. Você enxerga outra fonte para a solução de armazenamento de informações ?
7. Como você caracterizaria um "bom" resultado (saída) que seria gerado pôr uma solução bem-sucedida?
8. Que documentos você utiliza nas atividades: documentos, anotações, formulários, relatórios,
9. Você poderia mostrar-me (ou descrever-me) o ambiente em que a solução será usada?
10. Quais informações devem ser fornecidas?
11. Quais informações devem ser armazenadas?
12. Quais informações serão produzidas?
13. Que funções e desempenho são exigidos?
14. Existe tecnologia para construir o sistema?

Respostas FL01:

1. Coordenar Estágios e Visitas dos Alunos de graduação – Fátima – Prof. Lenarti – www.ceo.efei.br
2. Responsáveis:

2.1 Processo Seletivo – Vestibular – Catarina
2.2 Processo de transferência Interna e Externa – Catarina
2.3 Normas – Normalização – Catarina – Prof. Valadão – Olívia.
2.4 Controle de horários – Leo – Elisa.
2.5 Criação, autorização e reconhecimento de cursos de graduação – Catarina – Prof. Valadão.
2.6 Planejamento Educacional – Projeto.

Sugestão:
Prof. Valadão
Nova área a ser pesquisada: **Sistema de Controle de Informação do Docente**
Atualmente as informações estão repartidas entre DEP – ASP – CPPD
Estas informações são vitais para o Processo de avaliação de cursos
Processo de Reconhecimento de Cursos

3.2. ELABORAÇÃO DA PROPOSTA DE DESENVOLVIMENTO

Nesta fase, também chamada de fase de definição, inicia-se a etapa de planejamento do software. Durante essa etapa, uma descrição limitada do escopo do esforço de software é desenvolvida; uma análise de riscos é realizada, os recursos exigidos para se desenvolver o software são revistos; estimativas de custo e de prazo são estabelecidas [Pressman1995].

Neste item iremos apresentar uma ferramenta resultado da 1ª etapa do ciclo de vida de um software - a Proposta de Desenvolvimento. Esta Proposta faz parte da documentação do projeto, e visa

descrever a situação atual do Sistema, descrever o Sistema Proposto e permitir ao usuário a possibilidade de executar um estudo de viabilidade com base nas alternativas de implementação, benefícios e objetivos.

PROPOSTA DE DESENVOLVIMENTO

ÍNDICE

I. Objetivos do Documento

II. Histórico

1. Descrição do Sistema Atual

- **1.1. Objetivos**
- **1.2. Problemas Observados**
- **1.3. Importância do Sistema Atual**
- **1.4. Expectativa com o Processo de Automação**

2. Descrição do Sistema Proposto.

- **2.1. Objetivos**
- **2.2. Justificativas**
- **2.3. Premissas e Restrições**
- **2.4. Segurança**
- **2.5. Benefícios**

3. Viabilidade de Implementação

- **3.1. Risco Implementacional**
- **3.2. Alternativa de Implementação**
- **3.3. Segurança do Desenvolvimento**

4. Relação do Sistema com o Ambiente

- **4.1. Singularidade**
- **4.2. Integrabilidade**
- **4.3. Benefícios Globais**

5. Laudo da Apresentação

6. Anexos

I. Objetivos do documento

Este documento foi elaborado com o objetivo de descrever o atual funcionamento das atividades relativas ao levantamento de informações do Corpo Docente da EFEI, bem como apresentar uma proposta para o desenvolvimento de um novo Sistema de Informação, objetivando **aprimorar o sistema de gerenciamento de informações sobre o Corpo Docente.**

II. Histórico – dados da Instituição

Criada em **1913**, a partir da capacidade e do espírito empreendedor do seu fundador, **Theodomiro Carneiro Santiago**, com a finalidade de formar engenheiros capazes de se revelarem e de exercerem a engenharia **"mais por atos do que por palavras"**, o então **Instituto Eletrotécnico e Mecânico de Itajubá**, desde logo se destacou na formação de profissionais especializados em sistemas energéticos, especialmente em geração e transmissão de energia elétrica.

Em 30 de janeiro de 1956, já com a denominação de **Instituto Eletrotécnico de Itajubá**, foi federalizado pela Lei n.º 2.721. Teve sua denominação alterada para **Escola Federal de Engenharia de Itajubá** em 16 de abril de 1968, pelo Decreto n.º 62.567.

A competência e o renome adquiridos em suas áreas de atuação conduziram ao desdobramento do seu curso original em cursos independentes de **Engenharia Elétrica** e de **Engenharia Mecânica**, com destaque especial para as ênfases de **Eletrotécnica** e **Mecânica Plena**. Iniciou, em 1968, seus cursos de **Pós-Graduação**, com os **Mestrados em Engenharia Elétrica e Mecânica.**

Em resposta à evolução da tecnologia e à expansão das novas áreas contempladas pela Engenharia, a EFEI ampliou as suas ênfases em 1980, passando a incluir a de **Produção** - no curso de Engenharia Mecânica - e a de **Eletrônica** - no de Engenharia Elétrica.

Em 1998, dando prosseguimento a uma política de expansão capaz de oferecer um atendimento mais amplo e diversificado à demanda nacional e, sobretudo, regional, de formação de profissionais da área tecnológica, passou a oferecer sete novos cursos de graduação: **Engenharia Ambiental, Engenharia da Computação, Engenharia de Controle e Automação, Engenharia de Produção - Mecânica, Engenharia Hídrica, Ciência da Computação e Administração**, os dois últimos no período noturno.

A oferta de cursos de Pós-graduação também foi ampliada, existindo atualmente **Cursos de Especialização em Engenharia de Sistemas Elétricos, em Qualidade e Produtividade e em Informática Empresarial, Programas de Mestrado em Engenharia Elétrica, Engenharia da Energia, Engenharia Mecânica, Ciência dos Materiais e em Engenharia de Produção, e Programas de Doutorado em Engenharia Elétrica e em Engenharia Mecânica.**

A EFEI e suas fundações associadas atuam também, de forma intensa, há mais de vinte e cinco anos, na área de **educação continuada**, tendo já treinado mais de **cinqüenta mil profissionais de todas as regiões do País e também de países vizinhos**.

Consciente da importância de manter um corpo de professores altamente qualificado, de modo a sustentar e a aprimorar continuamente a qualidade, a atualidade e a atratividade de suas atividades de ensino, pesquisa e extensão, a EFEI investiu fortemente na capacitação dos seus docentes.

A organização didático-pedagógica da EFEI alinha-se às diretrizes defendidas por seu fundador, que defendia o espaço escolar como um espaço de estabelecimento de vinculação entre o saber e o fazer.

Missão Institucional: "Gerar, preservar e difundir conhecimento, formar cidadãos e profissionais qualificados, e contribuir para o desenvolvimento do país, visando a melhoria da qualidade da vida".

ÁREAS DE ATUAÇÃO

A EFEI se notabilizou no cenário da Engenharia nacional graças ao seu quadro de professores altamente qualificado, responsável pela formação acadêmica de seus alunos. Aproximadamente 70 laboratórios com tecnologia de ponta, implantados paulatinamente, também contribuíram para o reconhecimento da instituição. Todo esse esforço de expansão vem resultando na formação de vários profissionais de renome nacional e internacional que, atualmente, prestam seus serviços em diversos setores da vida nacional, a exemplo das áreas política e tecnológica. Em 1998, a EFEI deu um verdadeiro salto de qualidade com a implantação de novos cursos, a maioria voltados para a área tecnológica, sua vocação histórica. Atualmente a Escola dispõe de 20 cursos, 11 de graduação e 9 de pós-graduação, distribuídos entre o mestrado, o doutorado e a especialização. Segundo os princípios do "tripé" das universidades renomadas – o Ensino, Pesquisa e Extensão – a EFEI vem atuando fortemente na interação com a comunidade local, regional e nacional, através da prestação de serviços e da construção de parcerias com empresas de grande e médio portes do Brasil e do exterior, além de órgãos públicos, com vistas ao seu permanente desenvolvimento científico e tecnológico.

Seu espaçoso Campus conta com modernas e sofisticadas instalações propícias ao desenvolvimento das atividades acadêmicas e administrativas, onde agrega-se um moderno complexo poliesportivo. A EFEI conta, ainda, com um importante diferencial: o campus avançado conhecido como Centro de Inovações Tecnológicas em Energia e Meio Ambiente – Usina Hidroelétrica Luiz Dias – totalmente revitalizado, que vem sendo utilizado como laboratório vivo dos setores energético e ambiental na realização de aulas práticas, ensaios, pesquisas tecnológicas e cursos de extensão à comunidade.

1. Descrição do sistema Atual

1.1. Objetivos

Tentar levantar as informações do Corpo Docente da EFEI.

1.2. Problemas observados

Atualmente não existe uma sistemática para levantamento de informações dos docentes. O que existe são procedimentos isolados que cada departamento adota para levantamento das informações, causando muitas vezes ao docentes a necessidade de oferecer várias vezes a vários locais as mesmas informações, correndo o risco de repassar informações diferentes e desatualizadas.

1.3. Importância do Sistema atual

Apesar dos riscos do levantamento de informações desta forma, este serviço é indispensável para outros Sistemas.

1.4. Expectativa com o processo de automação

O processo de levantamento de informações se tornará mais rápido e mais preciso, evitando redundâncias e duplicações de informações.

2. Descrição do Sistema Proposto

2.1. Objetivos

a. Integrar as informações do CORPO DOCENTE da INSTITUIÇÃO hoje dispersas;
b. Evitar duplicação de coleta de dados;
c. Estimular a transparência da ação docente;
d. Permitir a avaliação do trabalho docente;
e. Facilitar as consultas dos usuários;
f. Economizar tempo e recursos;
g. Possibilitar a divulgação dos dados;
h. Garantir uniformidade e clareza nas informações;
i. Permitir levantamento de informações para fins de Criação, Aprovação e Renovação de Cursos da Instituição;
j. Promover a **atualização tecnológica** da Instituição no tocante a Análise de Modelagem de Dados com utilização de **ferramentas modernas** de desenvolvimento;
k. **Simplificação das rotinas administrativas**, agilizando a tramitação de documentos e informações, acelerando e aprimorando a qualidade dos serviços prestados pela Instituição;
l. **Redução de custos administrativos** por meio de:

- acesso a informações produzidas e mantidas em diferentes ambientes computacionais, inclusive de diferentes portes, por uma mesma máquina de pequeno porte;
- diminuição de uso de papel provocada pela automação e informatização das comunicações internas.
- Aumento da **eficiência administrativa** nas operações internas, em virtude de maiores facilidades de comunicação e da integração entre institutos e departamentos, possibilitando a implantação de produtos como correio eletrônico, agendas de compromissos, quadro de troca de idéias e muitos outros.

m. Tratamento de informação em tempo real.

2.2. Justificativas

Embora vivenciando a revolução provocada pela informática na área de comunicação, a EFEI, até o momento, não dispõe de uma administração automatizada, dependendo de **procedimentos administrativos manuais** e de alguns **sistemas de informação** com características **não integradas**.

Tais sistemas não eliminam as **rotinas manuais** conduzindo à superposição e **duplicação de tarefas**, comprometendo o processo decisório devido a manipulação de dados inconsistentes.

Assim sendo, a não adoção de um **modelo global de dados** auxiliado por um sistema integrado de informação conduz a uma falta de dados que impede uma rápida tomada de decisão.

Na página : http://www.mec.gov.br/sesu/ofertas.shtm pode-se avaliar a necessidade de todas as instituições se organizarem para atender as exigências legais que hoje se impõem. Dentre os itens avaliados em cada instituição a qualificação e o desempenho CORPO DOCENTE com certeza ocupam situação de destaque.

"A Avaliação das Condições de Oferta de Cursos de Graduação é uma ação da Secretaria de Educação Superior (SESu) que visa a avaliar, de acordo com o disposto na Lei nº 9.131, de 24 de novembro de 1995, Decreto nº 2.026, de 10 de outubro de 1996, e Lei nº 9.394, de 20 de dezembro de 1996, de Diretrizes e Bases da Educação Nacional, *in loco*, cada um dos cursos de graduação submetidos ao Exame Nacional de Cursos (Provão), com relação à **qualificação de seu corpo docente**, à sua organização didático-pedagógica e a suas

instalações, tanto as físicas em geral, quanto as especiais, tais como laboratórios, equipamentos e bibliotecas.

A avaliação periódica dos cursos e instituições de ensino superior, como determina a legislação, deve utilizar-se de procedimentos e critérios abrangentes com relação aos diversos fatores que determinam a qualidade e a eficiência das atividades de ensino, pesquisa e extensão."

Em nossa instituição os dados sobre o corpo docente são dispersos pelos diversos departamentos, não são atualizados com a freqüência necessária e são apresentados em diferentes modelos e configurações. Um sistema de informação unificado, eficiente, eficaz, atualizado facilitaria a vida de todos departamentos que prestam informações alem de proporcionar o acompanhamento, controle e avaliação constante da ação docente no ensino de graduação e pós-graduação, na pesquisa e na extensão.

2.3. Premissas e restrições

Este projeto esta sendo desenvolvido para fins acadêmicos, sem fins lucrativos para rodar em ambiente Cliente/Servidor. O desenvolvimento abrangerá recursos humanos, de software e de hardware limitados sob as seguintes condições:

i) Qualquer SO para o qual exista uma Java Virtual Machine poderá ser utilizado.
ii) A linguagem de codificação será JAVA e utilizaremos a ferramenta Visual Age.
iii) Para o desenvolvimento e utilização será exigida uma máquina com processador Celeron 300Mhz 128 MB de RAM.
v) O sistema necessitará de 3 pessoas para o seu desenvolvimento e o tempo estimado de desenvolvimento será de 14 semanas.
vi) A interface da aplicação será construída com base nas classes de objetos existentes na ferramenta de desenvolvimento e na própria linguagem.

2.4. Segurança

Futuramente serão estabelecidas normas de segurança para acesso aos dados.

2.5. Benefícios

Permitir a integração das informações do CORPO DOCENTE da INSTITUIÇÃO;

Estabelecer um Método para desenvolvimento de software;

Servir de base para desenvolvimentos futuros;

Estabelecer um formato de documentação para projetos de desenvolvimento de Banco de Dados;

Criar um ambiente físico bem estruturado para manter informações;

Criar um ambiente confiável para as informações;

Oferecer agilidade e rapidez no fornecimento de informações;

Manter uma Base de Dados permanente e atualizada.

3. Viabilidade de implementação

3.1. Risco implementacional (Risco de negócio, de projeto e técnico)

Não há risco implementacional, pois a ASP possui um SGBD, e o CPD possui ferramenta de desenvolvimento, bem como recurso computacionais e pessoal qualificado disponível..

3.2. Alternativas de implementação

1° alternativa de implementação:

3.2.1. *Marcos, pontos de controle*

Estudo Inicial – Estudo de Viabilidade – Coleta de Informações	- 21/11/00
Elaboração da Proposta de Desenvolvimento	- 21/11/00
Análise do Projeto Lógico de Banco de Dados	
Desenvolvimento do Modelo Conceitual:	
Modelo Entidade-Relacionamento (DER)	- 27/11/00
Projeto Físico: Conversão do Modelo E-R para	
o Modelo Relacional	- 11/12/00
Elaboração do Dicionário de Dados	- 18/12/00
Desenvolvimento dos Diagramas – UML – Rational Rose	- 29/12/00
Implementação de Banco de Dados Físico – SGBD	- 29/01/01
Desenvolvimento do Software – Visual Age	- 29/02/01
Geração de Testes de Aceitação/Avaliação de Riscos	- 29/03/01
Elaboração de um Sistema da Garantia da Qualidade para	
o desenvolvimento do BD	- 29/04/01

3.2.2. *Procedimentos de aprovação*

Assinatura no final deste documento, pelo aluno mestrando e pelo Prof. Orientador.

3.2.3. *Recursos Necessários*

3.2.3.1. *Cronograma de Recursos:*

Não será necessária a aquisição de hardware ou software, pois a UNIFEI possui uma Rede Local, computadores e software necessários para o desenvolvimento deste projeto. Se houver a necessidade de aquisição de hardware e software, devemos descrever os itens usando a tabela abaixo.

Mês Discriminação Valor

3.2.3.2. *Cronograma de Desembolso:*

O Cronograma de Recursos descreve a quantidade de horas técnicas trabalhadas e seu custo. Este projeto não terá custo de mão de obra. Se houver custo, devemos relaciona-los usando a tabela abaixo.

Mês Horas trabalhadas Preço R$

3.2.4. Orçamento

= Cronograma de Recursos + Cronograma de Desembolso

3.2.5. Equipe

O projeto será desenvolvido por um aluno de Mestrado em Eng. Elétrica. Se usássemos uma equipe a descrição deste item seria:

O projeto será desenvolvido por uma equipe de .... membros a taxa/hora de R$ ... por pessoa....

3.3. Segurança do desenvolvimento

Serão usadas cópias backups do projeto em HD's e Zip Drive.

4. Relação do Sistema com o ambiente

4.1. Singularidade

Existem soluções desenvolvidas por outras Universidades, mas não disponíveis para compra, utilização ou acesso ao projeto.

4.2. Integrabilidade

Este projeto propõe-se a desenvolver parte de uma Base de Dados Global de Informações, podendo futuramente integrar-se a outras Bases e/ou Sistemas.

4.3. Benefícios Globais

Integrar as informações do CORPO DOCENTE da INSTITUIÇÃO hoje dispersas, evitar duplicação de coleta de dados, estimular a transparência da ação docente, permitir a avaliação do trabalho docente, facilitar as consultas dos usuários, economizar tempo e recursos, possibilitar a divulgação dos dados, garantir uniformidade e clareza nas informações, promover a atualização tecnológica da Instituição no tocante a Análise de Modelagem de Dados com utilização de ferramentas modernas de desenvolvimento, simplificação das rotinas administrativas, agilizando a tramitação de documentos e informação, acelerando e aprimorando a qualidade dos serviços prestados pela Instituição, redução de custos administrativos, tratamento de informação em tempo real, permitir a integrar as informações do corpo docente da instituição, estabelecer um Método para desenvolvimento de modelos de dados, servir de base para desenvolvimentos futuros, estabelecer um formato de documentação para projetos de desenvolvimento de Banco de Dados, criar um ambiente físico bem estruturado para manter informações, criar um ambiente confiável para as informações, oferecer agilidade e rapidez no fornecimento de informações e manter uma Base de Dados permanente e atualizada.

5. Laudo da apresentação da proposta

Esta proposta será apresentada ao Prof. Orientador e aos usuários, e posteriormente será feito um laudo.

6. Anexos da proposta

Documentos adicionais, referências a metodologias, formulas, etc...

3.3. LEVANTAMENTO/ANÁLISE PRELIMINAR DE REQUISITOS(LPR)

3.3.1. MODELO DESCRITIVO(Descrição Narrativa)

Após a etapa de entrevistas e a elaboração da "Proposta de Desenvolvimento", foi criado o primeiro modelo da fase de Análise de requisitos, o modelo descritivo, que compreende uma narrativa pela visão do desenvolvedor de software, visando destacar os pontos principais do funcionamento do software.

Funcionamento do Sistema de Informação dos Docentes

O Sistema de Informação do Docente, deverá permitir consultar, alterar e acrescentar novos dados ao cadastro do professor tipo:

- informações sobre o seu login e senha, que valerá como entrada para o sistema de acesso;
- dados pessoais, bem como departamento no qual está lotado;
- permitir registrar trabalhos em congressos, publicações, capacitações, portarias, progressões funcionais, incentivo funcional, participação em processo seletivo, bancas examinadoras, comissões e pesquisas ;
- permitir seu enquadramento em áreas e sub áreas do conhecimento humano;
- registrar as preferências e não preferências de disciplinas, horários, espaços físicos(laboratórios) e recursos utilizados;
- permitir a atualização de seu currículo possibilitando sua importação ou exportação (modelo CAPES/CNPq);
- permitir ao professor o planejamento das atividades do ano, através de um sistema de agenda;
- Permitir levantamento de informações para fins de Criação, Aprovação e Renovação de Cursos da Instituição;

3.4. REQUISITOS DO SISTEMA

[Pressman1995] cita no capítulo 5, que há 450 anos, Maquiavel disse: "...não há nada mais difícil de se tomar nas mãos, mais perigoso de se conduzir ou mais incerto quanto ao seu sucesso do que a iniciativa de se introduzir uma nova ordem de coisas...".

Neste item, procuraremos identificar com base no modelo descritivo, as entidades que farão parte das responsabilidades do sistema. O Sistema informatizado de Informação do Docente se caracteriza por conter as seguintes entidades distintas, a saber:

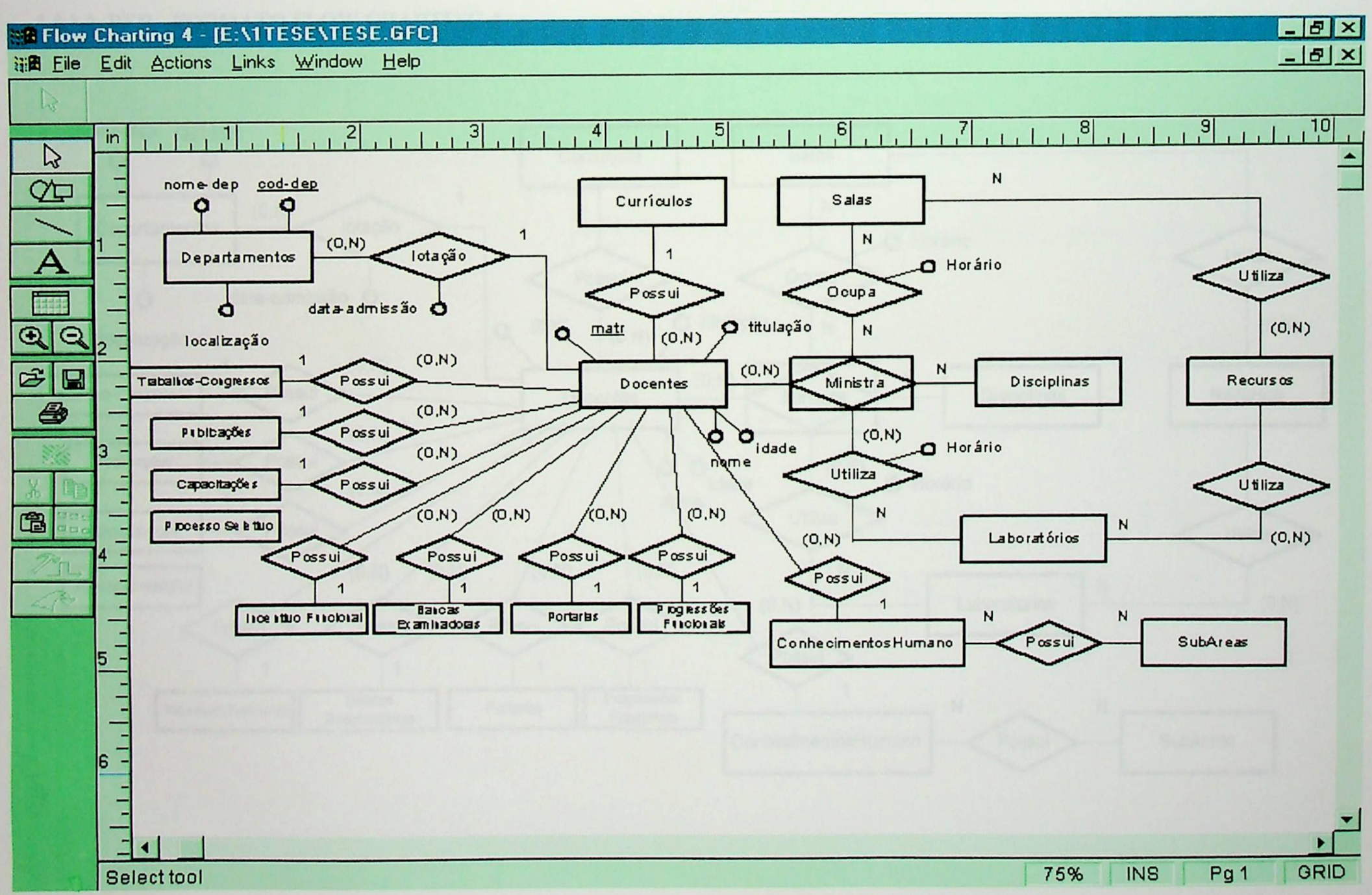
1. Docentes
2. Senhas
3. Departamentos
4. Currículos
5. Trabalhos-Congressos
6. Publicações-Docentes
7. Capacitações-Docentes
8. Portarias
9. Progressões-Funcionais
10. Incentivos-Funcionais
11. Processos-Seletivos
12. Bancas_Examinadoras-Comissões-Pesquisas
13. Conhecimento-Humano
14. Subareas
15. Disciplinas
16. Salas
17. Recursos
18. Laboratórios

3.5. ANÁLISE DO PROJETO LÓGICO DE BANCO DE DADOS

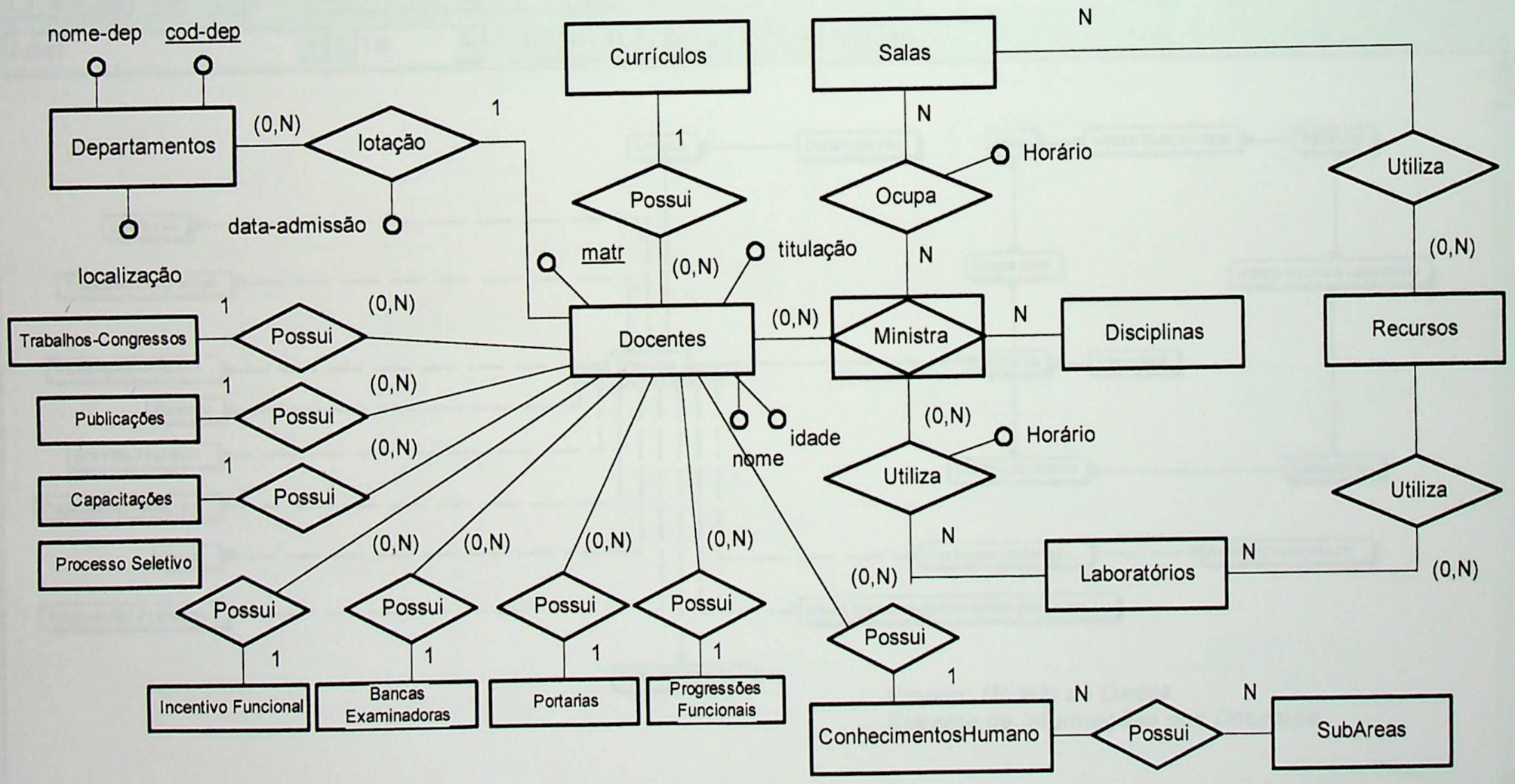
3.5.1. DESENVOLVIMENTO DO MODELO CONCEITUAL: MODELO ENTIDADE-RELACIONAMENTO (DER)

Neste item iremos representar toda nossa análise feita nas etapas anteriores (entrevistas, proposta de desenvolvimento, modelo descritivo), construindo o Modelo Conceitual do Sistema e o Modelo Relacional Normalizado. O modelo conceitual Relacional, visa representar a parte estrutural (aspectos estáticos) dos dados e seus relacionamentos. Utilizaremos uma ferramenta de diagramação "Flow Charting 4" para o desenvolvimento do MCR (modelo conceitual relacional). O modelo relacional normalizado, visa representar as ligações que ocorrerão entre as entidades, através da exportação das chaves primárias para as entidades relacionadas. Utilizaremos a ferramenta Case "Logic Works Erwin_ERX3.0" para desenvolver o modelo relacional normalizado. A seguir apresentamos os modelos nas duas ferramentas.

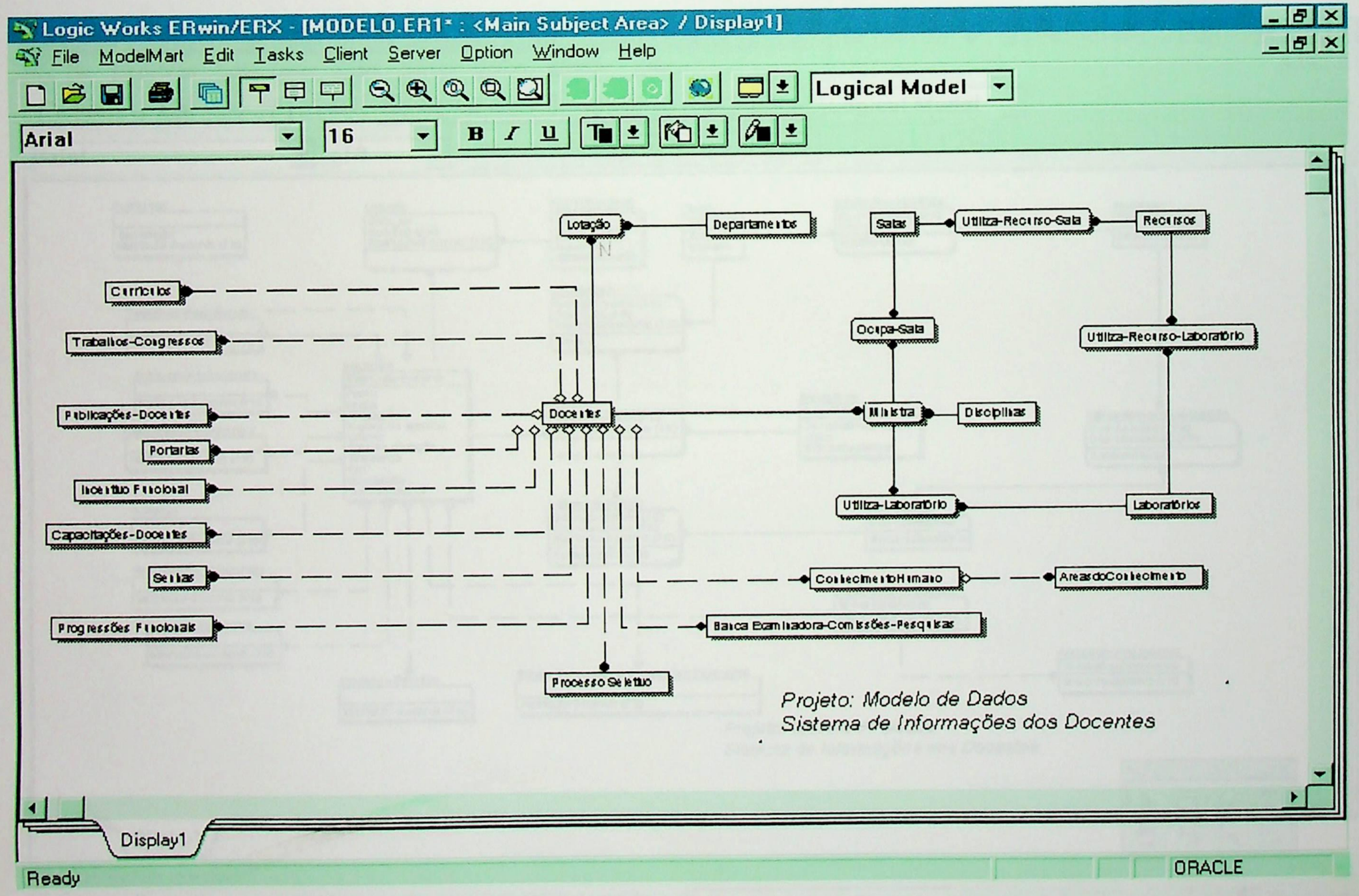
3.5.1.1. DER – FERRAMENTA: FLOW CHARTING 4



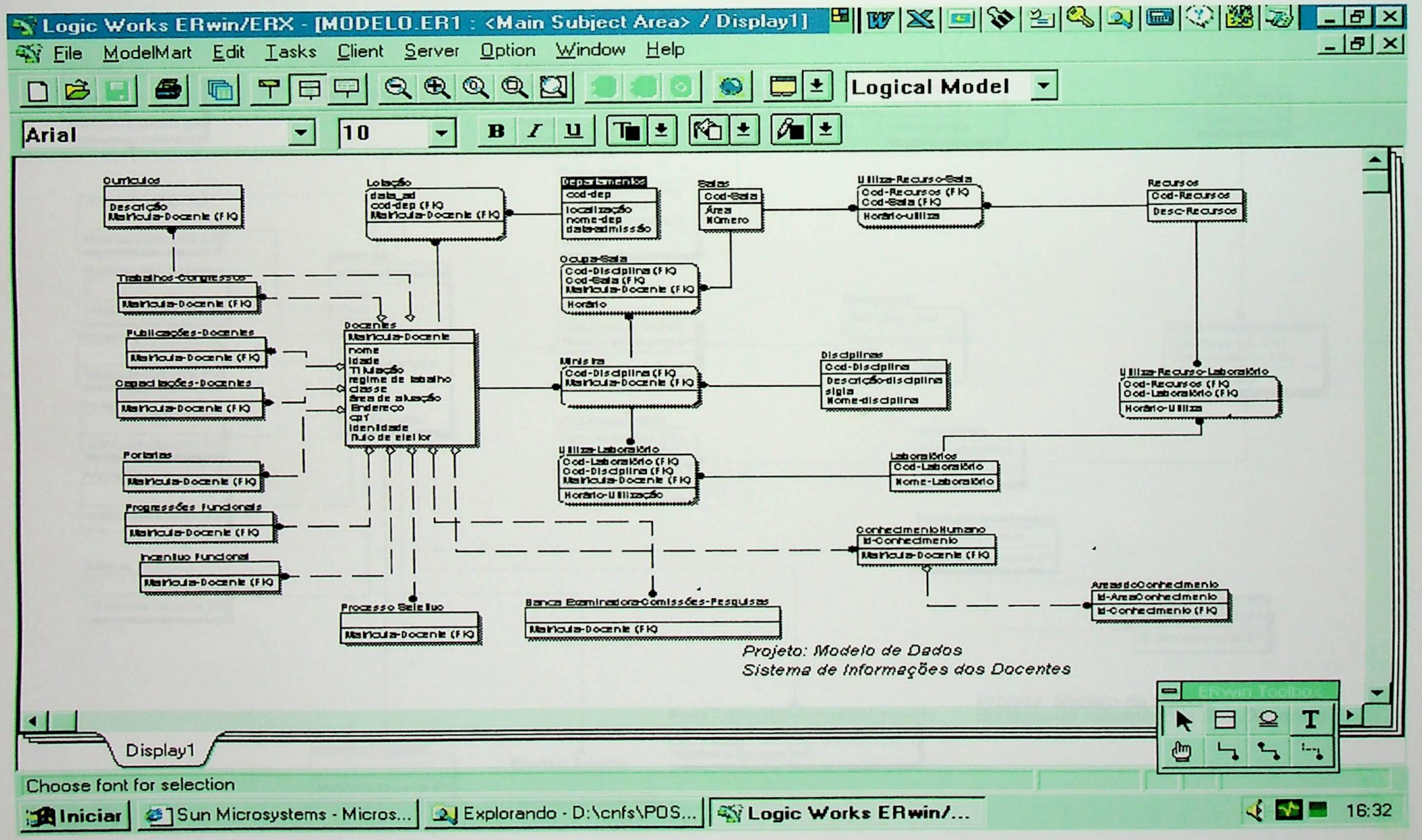
3.5.1.2. DER - FORMATO FLOW CHARTING 4



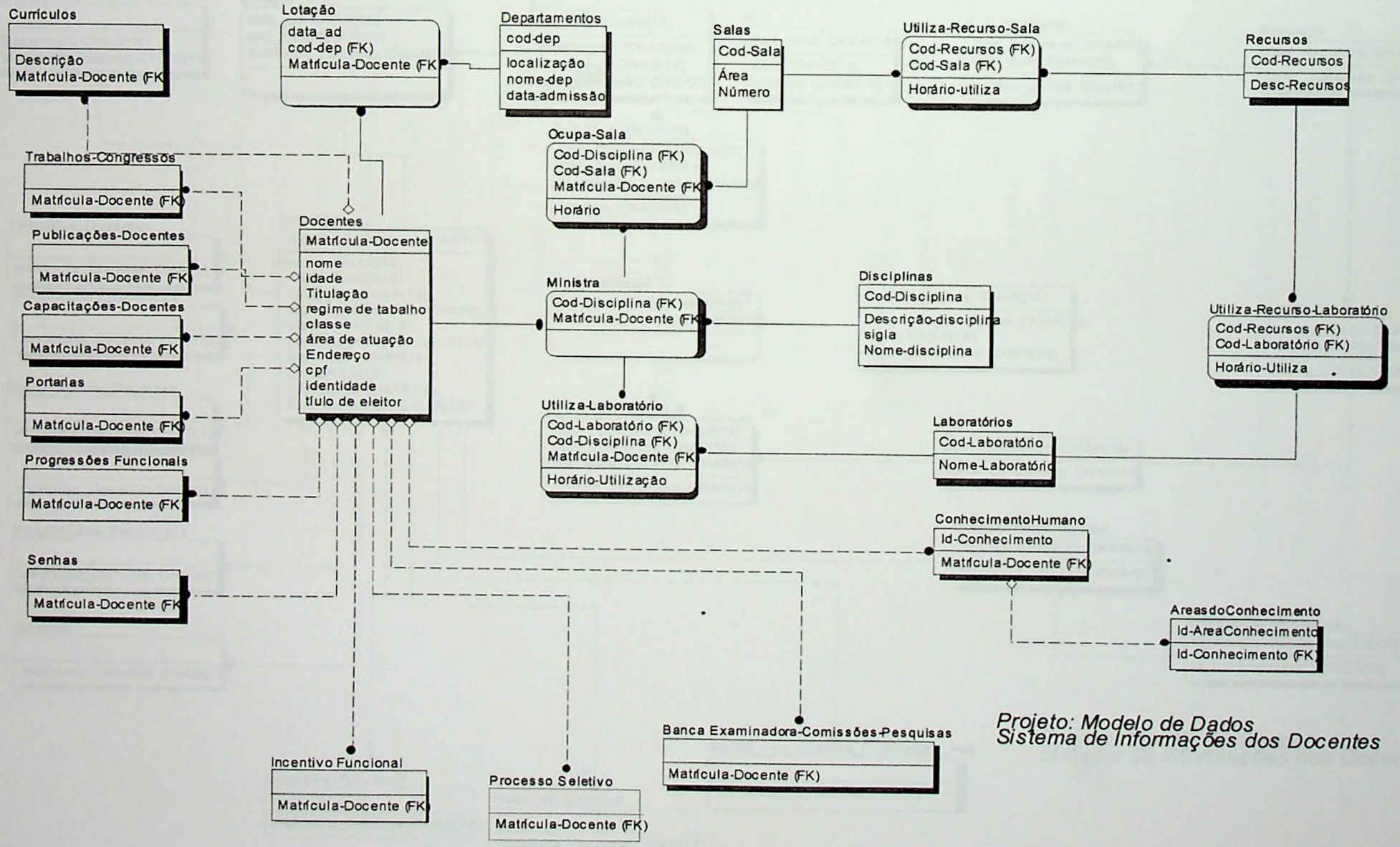
3.5.1.3. DER - FERRAMENTA LOGIC WORKS ERWIN_ERX3.0



3.5.1.4. DER – FORMATO LOGIC WORKS ERWIN_ERX3.0



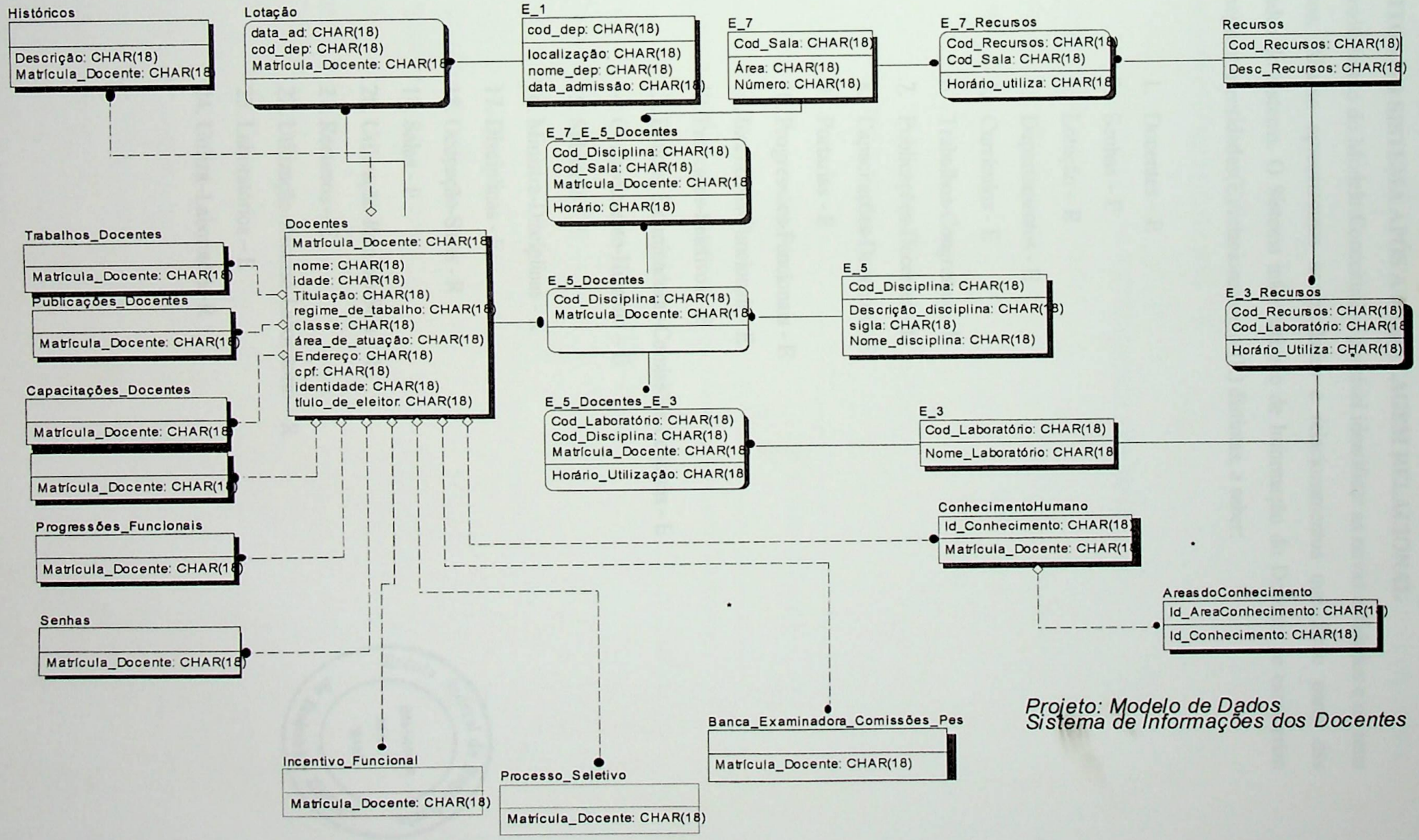
3.5.1.4. DER – FORMATO LOGIC WORKS ERWIN_ERX3.0 – Cont.



Projeto: Modelo de Dados
Sistema de Informações dos Docentes

3.6. PROJETO FÍSICO

3.6.1. MODELO RELACIONAL NORMALIZADO



Projeto: Modelo de Dados
Sistema de Informações dos Docentes

3.7. REQUISITOS DO SISTEMA APÓS A MODELAGEM RELACIONAL

Após o desenvolvimento do Modelo Conceitual, é possível identificar as novas entidades e os seus relacionamentos. Abaixo apresentamos as entidades e relacionamentos que farão parte das responsabilidades do sistema. O Sistema informatizado de Informação do Docente se caracteriza por conter as seguintes entidades(E)/relacionamentos(R) distintas, a saber:

1. Docentes – E
2. Senhas – E
3. Lotação – R
4. Departamentos - E
5. Currículos - E
6. Trabalhos-Congressos - E
7. Publicações-Docentes - E
8. Capacitações-Docentes - E
9. Portarias - E
10. Progressões-Funcionais - E
11. Incentivos-Funcionais - E
12. Processos-Seletivos - E
13. Bancas_Examinadoras-Comissões-Pequisas - E
14. Conhecimento-Humano – E
15. Subareas - E
16. Ministra-Disciplinas - R
17. Disciplinas - E
18. Ocupação-Salas - R
19. Salas - E
20. Utilização-Recurso-Sala - R
21. Recursos – E
22. Utilização-Recurso-Laboratório – R
23. Laboratórios – E
24. Utiliza–Laboratório-R

3.8. ELABORAÇÃO DO DICIONÁRIO DE DADOS -ENTIDADES/RELACIONAMENTOS

O Dicionário de Dados é uma ferramenta que visa descrever a especificação detalhada da entidade e de seus atributos, bem como sua restrição de acesso.

MODELO DE DICIONÁRIO DE DADOS

DESCRIÇÃO DA ENTIDADE/RELACIONAMENTO

PROJETO: Sistema de Informação do Docente

EQUIPE: Coordenador: Cláudio Neves Sá DATA: 08/06/01

Analistas: VERSÃO:1.0

PÁGINA:01

ENTIDADE/RELACIONAMENTO

NOME: **Docentes** CÓDIGO: **Ent01**

DEFINIÇÃO: Esta entidade representa as informações cadastrais/pessoais do docente.

ATRIBUTOS: RESPONSÁVEL:

Matrícula-Docente (até 20 caracteres);
Nome (até 20 caracteres);
Idade (até 2 algarismos);
Titulação (até 20 caracteres);
Regime de trabalho (até 20 caracteres);
Classe (até 20 caracteres);
Área de atuação (até 20 caracteres);
Endereço (até 20 caracteres);
CPF (até 20 caracteres);
Identidade (até 20 caracteres);
Titulo de Eleitor (até 20 caracteres);
Fone Comercial (até 14 algarismos);
Ramal (até 5 algarismos);
Fax Comercial (até 14 algarismos);
E-mail Profissional (até 35 caracteres);

Home Page da Empresa (até 50 caracteres – suprimindo o prefixo http://);

Data do Aniversário (no formato DD/MM – Dia e Mês – até 5 caracteres);

Nome do Cônjuge (até 35 caracteres).

IDENTIFICADORES: Matrícula-Docente

DEPENDÊNCIA DA EXISTÊNCIA DA ENTIDADE:

DEPENDÊNCIA FUNCIONAL ENTRE ATRIBUTOS:

Todos os atributos dependem funcionalmente de Matrícula-Docente

DIMENSÃO:

ENTIDADES/AGR.	PAPEL	CONECTIVIDADE	PARTICIPAÇÃO

CARDINALIDADE:

Mínima: Média: Máxima: Crescimento:

RESTRIÇÕES DE INTEGRIDADE:

RESTRIÇÕES DE ACESSO:

Classes de Usuários:	C	R	U	D
Docentes	X	X	X	
DEP – ASP – CPPD - PDG	X	X	X	
Desenvolvedores de Software		X		
Administradores do Sistema	X	X	X	X
Usuários Intranet/Internet		X		

OBS:

3.9. CLIENTES DO SISTEMA – CLASSES DE USUÁRIOS:

Com base nas etapas anteriores foram identificados os seguintes clientes/atores do Sistema. Esta descrição será utilizada na representação do Diagrama da UML que visa representar a parte dinâmica do Sistema.

Docentes: É o maior beneficiário do Sistema, portanto, tem acesso completo ao sistema, podendo incluir, alterar, excluir e consultar seus dados.

DEP – ASP – CPPD – PDG:

Desenvolvedores de Software:

Administradores do Sistema:

Usuários Intranet/Internet:

3.10. FUNÇÕES DO SISTEMA

Uma operação muda um objeto de alguma forma. Mais especificamente, ela muda um ou mais valores de atributo que estão contidos no objeto. Por conseguinte, uma operação deve ter "conhecimento" da natureza dos atributos do objeto e deve ser executada de uma forma que a capacite a manipular as estruturas de dados que foram derivadas dos atributos.

Embora existam muitos tipos diferentes de operações, geralmente elas podem ser divididas em três categorias: operações de manipulação, de computação e monitoração de um objeto quanto à ocorrência de um evento controlador [Pressman1995].

Neste item iremos relacionar as funções do Sistema, visando o seu comportamento.

- F0 – Checar login/Senha;
- F1 - Cadastrar/Alterar/Excluir Login e Senha;
- F2 - Cadastrar/Alterar/Excluir dados pessoais, bem como departamento ao qual está lotado;
- F3 - Cadastrar/Alterar/Excluir: Trabalhos em congressos, publicações, capacitações, portarias, progressões funcionais, incentivo funcional, participação em processo seletivo, bancas examinadoras, comissões e pesquisas ;
- F4 - Cadastrar/Alterar/Excluir o enquadramento do docente em áreas e sub áreas do conhecimento humano;
- F5 - Cadastrar/Alterar/Excluir as preferências e não preferências de disciplinas, horários, espaços físicos(laboratórios) e recursos utilizados;
- F6 - Cadastrar/Alterar/Excluir o currículo do docente;

- F7 - Cadastrar/Alterar/Excluir o planejamento das atividades do ano, através de um sistema de agenda;
- F8 - Possibilitando a importação ou exportação do currículo (modelo CAPES/CNPq);
- F9 - Consultar informações para fins de Criação, Aprovação e Renovação de Cursos da Instituição;
- F10 - Sair do Sistema.

3.11. DIAGRAMAS UML – FERRAMENTA: RATIONAL ROSE/C++ DEMO

Na criação dos modelos baseados em diagramas UML, usaremos a ferramenta Rational Rose para representação da parte estrutural estática e da parte dinâmica do Sistema.

Diagrama de Classes

Diagrama de Objetos

Diagrama de Componente

Diagrama de Execução

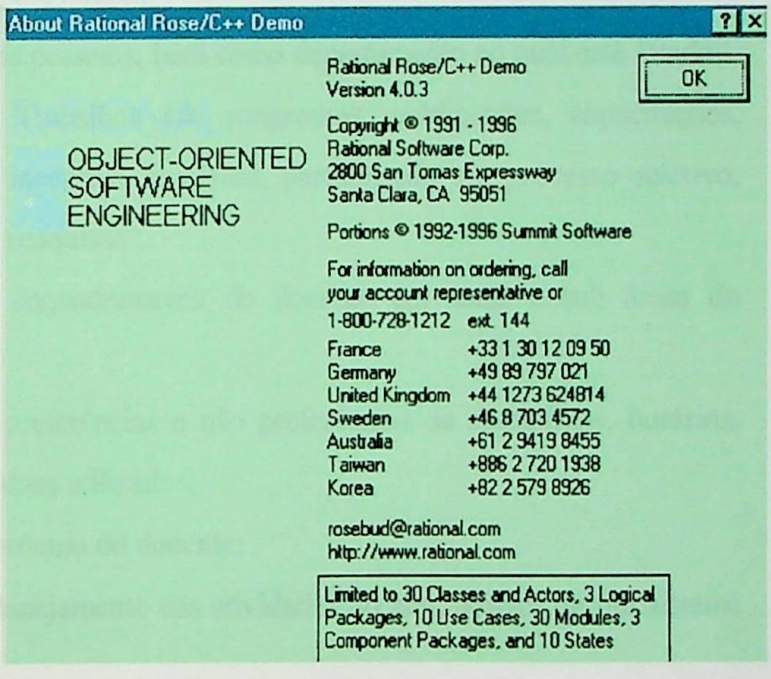
Diagrama Use-Case

Diagrama de Seqüência

Diagrama de Estado

Diagrama de Colaboração

Diagrama de Atividade



3.12. CENÁRIOS USE-CASE

Nenhum sistema existe isoladamente [Booch2000]. Todo sistema interessante interage com atores humanos ou autômatos que utilizam esse sistema para algum propósito e esses atores esperam que o sistema se comporte de acordo com as maneiras previstas. Um caso de uso especifica o comportamento de um sistema ou de parte de um sistema e é uma descrição de um conjunto de seqüências de ações, incluindo variantes realizadas pelo sistema para produzir um resultado observável do valor do ator. Os casos de usos podem ser aplicados para captar o comportamento pretendido do sistema que está sendo desenvolvido. A seguir descrevemos algumas funções exigidas pelos atos descritos no item 3.9 – Clientes do Sistema.

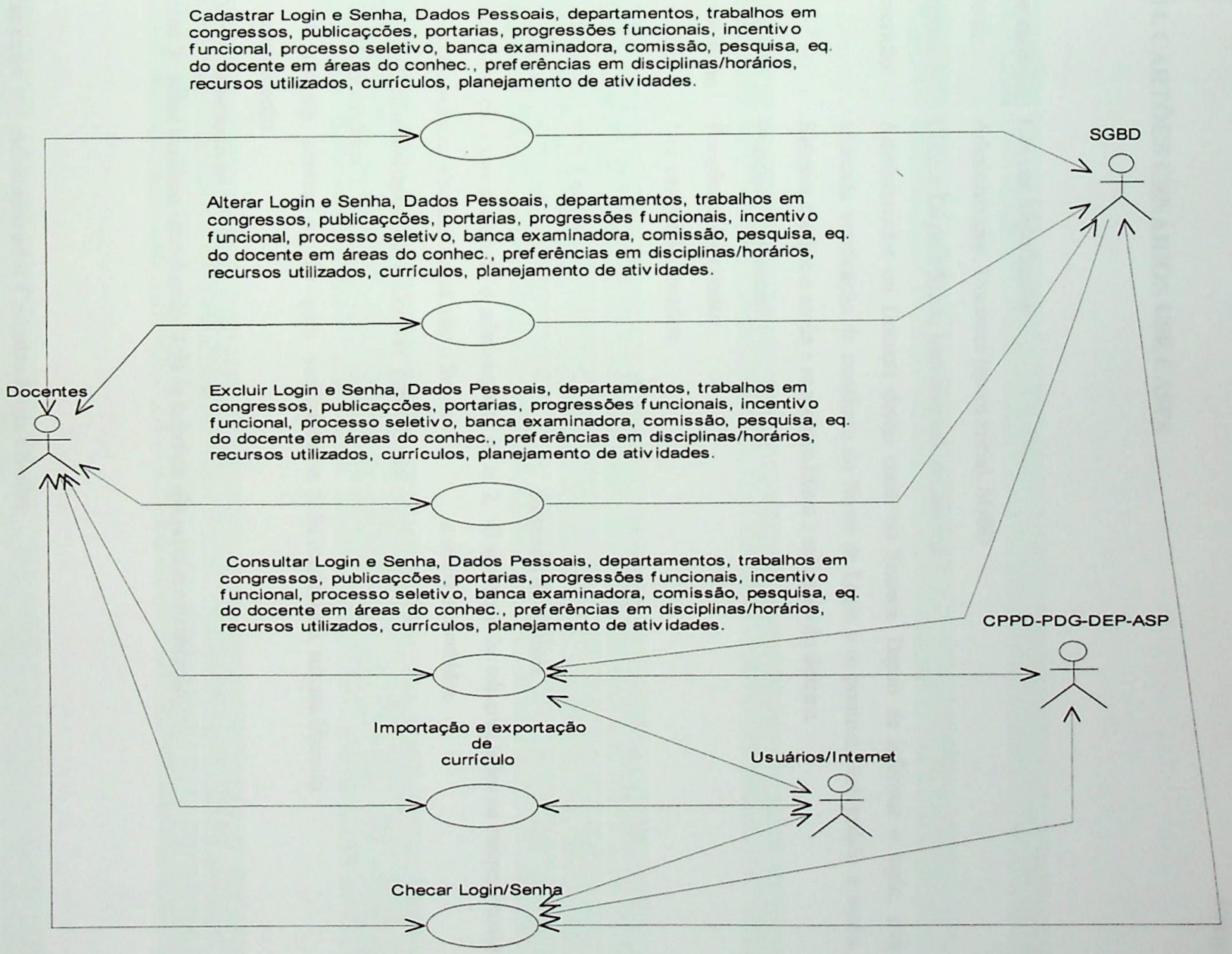
- F0 – Checar login/Senha;
- F1 - Cadastrar/Alterar/Excluir Login e Senha;
- F2 - Cadastrar/Alterar/Excluir dados pessoais, bem como departamento ao qual está lotado;
- F3 - Cadastrar/Alterar/Excluir: Trabalhos em congressos, publicações, capacitações, portarias, progressões funcionais, incentivo funcional, participação em processo seletivo, bancas examinadoras, comissões e pesquisas ;
- F4 - Cadastrar/Alterar/Excluir o enquadramento do docente em áreas e sub áreas do conhecimento humano;
- F5 - Cadastrar/Alterar/Excluir as preferências e não preferências de disciplinas, horários, espaços físicos(laboratórios) e recursos utilizados;
- F6 - Cadastrar/Alterar/Excluir o currículo do docente;
- F7 - Cadastrar/Alterar/Excluir o planejamento das atividades do ano, através de um sistema de agenda;
- F8 - Possibilitando a importação ou exportação do currículo (modelo CAPES/CNPq);
- F9 - Consultar informações para fins de Criação, Aprovação e Renovação de Cursos da Instituição;
- F10 - Sair do Sistema.

3.13. DIAGRAMA USE-CASE

(Visão de caso de uso)

[Booch1997] A modelagem de um diagrama use-case é uma técnica usada para descrever e definir os requisitos funcionais de um sistema. Eles são escritos em termos de atores externos, use-cases e o sistema modelado. Os atores representam o papel de uma entidade externa ao sistema como um usuário, um hardware, ou outro sistema que interage com o sistema modelado.

Um *diagrama de caso de uso* mostra um conjunto de casos de uso e atores (um tipo especial de classe) e seus relacionamento.



3.14. CARTÕES CENÁRIOS USE CASES

0

Use case:	**Checar login/Senha**
Atores:	Administrador, Docentes (quem inicia), SGBD
Purpose:	Checar Login e Senha para liberação do sistema.
Descrição:	Administrador ou Docente deseja entrar no Sistema. Depois de informar o login, será efetuada verificação da existência do Nome de Login e se positivo, será pedido a senha. Sistema confere a senha e se positivo libera a utilização do Sistema.
Type	Principal e essencial
Referências Cruzadas	Funções relacionadas: Use cases relacionados:

Curso de Eventos Típicos	
Ação do Ator	**Resposta do Sistema**
1. Este use case inicia quando o administrador ou o docente solicita liberação para usar o Sistema.	2. O sistema exibe tela com todos os campos vazios a serem preenchidos.
3. O administrador ou o docente preenche os campos necessários.	
4. O Sistema acessa o BD para verificação e autenticação.	5. Se confirmado, sistema liberado.

Cursos Alternativos
Linha 3: valor inválido introduzido. Tela da Interface não permite autenticação.

1

Use case:	**Administrador Cadastra Login e Senha**
Atores:	Administrador (quem inicia), SGBD
Purpose:	Efetuar cadastro de Login e Senha
Descrição:	Administrador deseja incluir um novo login/senha. Depois de efetuar a inserção dos dados

	do novo login e senha, solicita o cadastro. Sistema verifica e cadastra o login e senha.
Type	Principal e essencial
Referências Cruzadas	Funções relacionados: Use cases relacionados:

Curso de Eventos Típicos

Ação do Ator	**Resposta do Sistema**
1. Este use case inicia quando o administrador solicita a inclusão de um novo login e senha na tela de login/senha pressionado o botão Cadastrar.	2. O sistema exibe tela com todos os campos vazios a serem preenchidos.
3. O administrador preenche os campos necessários.	
4. Ao término da entrada do login e senha, o administrador solicita que a informação seja armazenada clicando no Botão Salvar.	5. Se não houver coincidência o login e senha são armazenados.

Cursos Alternativos

Linha 3: valores inválido introduzido. Tela da Interface não permite Inclusão.

Linha 4: Administrador não confirma inclusão pressionando botão Cancelar da tela de login/senha. Retorna a tela de login/senha apagando o que já foi preenchido.

Linha 5: O administrador tenta incluir um login sem senha associada. O sistema não habilita o botão salvar.

2

Use case:	**Administrador/Docente Altera Senha**
Atores:	Administrador/Docentes (quem inicia), SGBD
Purpose:	Efetuar alteração de Senha do Login no BD.
Descrição:	Administrador/Docente deseja alterar uma senha. Depois de efetuar a alteração dos dados da nova senha, solicita a alteração. Sistema verifica e altera a senha.
Type	Principal e essencial
Referências Cruzadas	Funções relacionadas: Use cases relacionados:

Curso de Eventos Típicos	
Ação do Ator	**Resposta do Sistema**
1. Este use case inicia quando um Administrador/Docente solicita a alteração de uma senha na tela de login/senha pressionando o botão Alterar.	2. O sistema exibe tela com todos os campos vazios a serem preenchidos.
3. O Administrador/Docente fornece o Login e a nova senha, redigita a nova senha.	
4. Ao término da entrada do login e senha, o administrador/Docente solicita que a informação seja armazenada clicando no Botão Salvar.	5. Se não houver inconsistência no dado, a senha é armazenada.
Cursos Alternativos	
Linha 3: valores inválido introduzido. Tela da Interface não permite alteração.	
Linha 4: Usuário não confirma alteração pressionando botão Cancelar da tela de login/senha. Retorna a tela de login/senha apagando o que já foi preenchido.	
Linha 5: O usuário tenta alterar uma senha sem um login associado. O sistema não habilita o botão alterar.	

3

Use case:	**Administrador Exclui Login**
Atores:	Administrador (quem inicia), SGBD
Purpose:	Efetuar Exclusão de Login no BD.
Descrição:	Administrador deseja excluir um login. Ele seleciona um login na tela de Login/Senha e pressiona o botão Excluir. O Sistema solicita confirmação da exclusão do Login e, em caso afirmativo, o exclui.
Type	Principal e essencial
Referências Cruzadas	Funções relacionadas: Use cases relacionados:

Curso de Eventos Típicos	
Ação do Ator	**Resposta do Sistema**
1. Este use case inicia quando o administrador seleciona um login na tela de login/senha.	2. O sistema exibe tela o login selecionado.
3. O administrador confere o(s) item(ns). Em seguida aciona o botão Excluir.	4. O sistema exibe uma tela com uma mensagem solicitando confirmação da exclusão.
5. Usuário confirma exclusão do login.	6. O login é excluído do armazenamento persistente.
Cursos Alternativos	
Linha 3: Item inválido introduzido. Tela da Interface não permite.	
Linha 5: Usuário não confirma exclusão. Sistema cancela transação de exclusão de Docente e retorna a tela de login/senha mantendo os dados do login selecionado.	

4

Use case:	**Administrador Consulta Login's**
Atores:	Usuário (quem inicia), SGBD
Purpose:	Efetuar consulta a login's no BD.
Descrição:	Usuário deseja consultar os login's.
Type	Principal e essencial
Referências Cruzadas	Funções relacionados: Use cases relacionados:

Curso de Eventos Típicos	
Ação do Ator	**Resposta do Sistema**
1. Este use case inicia quando o administrador solicita consulta a login's.	2. O sistema exibe tela com os campos
3. O administrador preenche o login a ser pesquisado ou * para visualizar todos os login's.	4. O sistema efetua pesquisa.
	5. O sistema exibe na tela todos os login's ou os dados do login pesquisado.
6. Usuário seleciona um Docente.	7. O sistema exibe Docente de forma detalhada.

Cursos Alternativos
Linha 3: Item inválido introduzido. Tela da Interface não permite.
Linha 6: Usuário pressiona botão finalizar para apagar parâmetros de consulta e visualizar todos os Docentes novamente.

3.15. CLASSES CANDIDATAS DA APLICAÇÃO

Com base no Diagrama Entidade-Relacionamento e Use Case é feito um refinamento, objetivando identificar as classes candidatas. As classes identificadas são as seguintes:

1. Docentes
2. Senhas
3. Lotação
4. Departamentos
5. Currículos
6. Trabalhos-Congressos
7. Publicações-Docentes
8. Capacitações-Docentes
9. Portarias
10. Progressões-Funcionais
11. Incentivos-Funcionais
12. Processos-Seletivos
13. Bancas_Examinadoras-Comissões-Pequisas
14. Conhecimento-Humano
15. Ministra-Disciplinas
16. Disciplinas
17. Ocupação-Salas
18. Salas
19. Utilização-Recurso-Sala
20. Recursos
21. Utilização-Recurso-Laboratório
22. Laboratórios
23. Utiliza –Laboratório

3.16. LEVANTAMENTO DE CLASSES DE INTERFACES COM O USUÁRIO

Analisando as funções e os atores do Diagrama Use Case, identificamos as possíveis classes de interface com o usuário. Cada classe de interface identificada, se constituirá posteriormente em uma janela do manual do usuário. O conjunto de classes é listado a seguir:

1. Tela de Docentes
2. Tela de Senhas
3. Tela de Lotação
4. Tela de Departamentos
5. Tela de Currículos
6. Tela de Trabalhos-Congressos
7. Tela de Publicações-Docentes
8. Tela de Capacitações-Docentes
9. Tela de Portarias
10. Tela de Progressões-Funcionais
11. Tela de Incentivos-Funcionais
12. Tela de Processos-Seletivos
13. Tela de Bancas_Examinadoras-Comissões-Pequisas
14. Tela de Conhecimento-Humano
15. Tela de Ministra-Disciplinas
16. Tela de Disciplinas
17. Tela de Ocupação-Salas
18. Tela de Salas
19. Tela de Utilização-Recurso-Sala
20. Tela de Recursos
21. Tela de Utilização-Recurso-Laboratório
22. Tela de Laboratórios
23. Tela de Utiliza –Laboratório

3.17. CLASSES DE PERSISTÊNCIA

Para cada classe de aplicação foi criada uma classe de persistência para que fosse possível guardar e ler os dados. Estas classes possuem funções de interação com SGBD (Sistema Gerenciador de Banco de Dados).

1. DocentesPersistência
2. SenhasPersistência
3. LotaçãoPersistência
4. DepartamentosPersistência
5. CurrículosPersistência
6. Trabalhos-CongressosPersistência
7. Publicações-DocentesPersistência
8. Capacitações-DocentesPersistência
9. PortariasPersistência
10. Progressões-FuncionaisPersistência
11. Incentivos-FuncionaisPersistência
12. Processos-SeletivosPersistência
13. Bancas_Examinadoras-Comissões-PequisasPersistência
14. Conhecimento-HumanoPersistência
15. Ministra-DisciplinasPersistência
16. DisciplinasPersistência
17. Ocupação-SalasPersistênciaPersistência
18. SalasPersistência
19. Utilização-Recurso-SalaPersistência
20. RecursosPersistência
21. Utilização-Recurso-LaboratórioPersistência
22. LaboratóriosPersistência
23. Utiliza –LaboratórioPersistência

3.18. LISTA DE RESPONSABILIDADES DA APLICAÇÃO

A partir da lista de verbos, leitura atenta da descrição narrativa e tendo como base o diagrama de Use Case, obtivemos as seguintes responsabilidades:

- F0 – Checar login/Senha;
- F1 - Cadastrar/Alterar/Excluir Login e Senha;
- F2 - Cadastrar/Alterar/Excluir dados pessoais, bem como departamento ao qual está lotado;
- F3 - Cadastrar/Alterar/Excluir: Trabalhos em congressos, publicações, capacitações, portarias, progressões funcionais, incentivo funcional, participação em processo seletivo, bancas examinadoras, comissões e pesquisas ;
- F4 - Cadastrar/Alterar/Excluir o enquadramento do docente em áreas e sub áreas do conhecimento humano;
- F5 - Cadastrar/Alterar/Excluir as preferências e não preferências de disciplinas, horários, espaços físicos(laboratórios) e recursos utilizados;
- F6 - Cadastrar/Alterar/Excluir o currículo do docente;
- F7 - Cadastrar/Alterar/Excluir o planejamento das atividades do ano, através de um sistema de agenda;
- F8 - Possibilitando a importação ou exportação do currículo (modelo CAPES/CNPq);
- F9 - Consultar informações para fins de Criação, Aprovação e Renovação de Cursos da Instituição;
- F10 - Sair do Sistema.

3.19. CARTÕES DAS CLASSES DE APLICAÇÃO

<table>
<tr><td colspan="2">Classe: Docentes
Descrição: Classe responsável pelo cadastro dos Docentes com suas informações e dados pessoais.</td></tr>
<tr><td>Responsabilidades identificadas:

• Cadastrar docentes
• Alterar docentes
• Excluir docentes</td><td>Colaborações:
DocentePersistência</td></tr>
</table>

• Consultar docentes	

Classe: Senhas	
Descrição: Classe responsável pelo cadastro das Senhas..	
Responsabilidades identificadas: • Cadastrar senhas • Alterar senhas • Excluir senhas • Consultar senhas	**Colaborações:** SenhaPersistência

3.20. LISTA DE RESPONSABILIDADES DAS CLASSES DE INTERFACES

Neste item serão descritas as funções que normalmente estão associadas as classes de Interfaces. Estas funções são comuns a todas as telas do Sistema.

- Exibir Tela de Docentes, Senhas, Lotação, Departamentos, Currículos, Trabalhos-Congressos, Publicações-Docentes, Capacitações-Docentes, Portarias, Progressões-Funcionais, Incentivos-Funcionais, Processos-Seletivos, Bancas_Examinadoras-Comissões-Pequisas, Conhecimento-Humano, Ministra-Disciplinas, Disciplinas, Ocupação-Salas, Salas, Utilização-Recurso-Sala, Recursos, Utilização-Recurso-Laboratório, Laboratórios, Utiliza –Laboratório.
- Fechar Tela de Docentes, Senhas, Lotação, Departamentos, Currículos,Trabalhos-Congressos, Publicações-Docentes, Capacitações-Docentes, Portarias, Progressões-Funcionais, Incentivos-Funcionais, Processos-Seletivos, Bancas_Examinadoras-Comissões-Pequisas, Conhecimento-Humano, Ministra-Disciplinas, Disciplinas, Ocupação-Salas, Salas, Utilização-Recurso-Sala, Recursos, Utilização-Recurso-Laboratório, Laboratórios, Utiliza –Laboratório.
- Cadastrar Tela de Docentes, Senhas, Lotação, Departamentos, Currículos,Trabalhos-Congressos, Publicações-Docentes, Capacitações-Docentes, Portarias, Progressões-

Funcionais, Incentivos-Funcionais, Processos-Seletivos, Bancas_Examinadoras-Comissões-Pequisas, Conhecimento-Humano, Ministra-Disciplinas, Disciplinas, Ocupação-Salas, Salas, Utilização-Recurso-Sala, Recursos, Utilização-Recurso-Laboratório, Laboratórios, Utiliza –Laboratório.

- Alterar Tela de Docentes, Senhas, Lotação, Departamentos, Currículos,Trabalhos-Congressos, Publicações-Docentes, Capacitações-Docentes, Portarias, Progressões-Funcionais, Incentivos-Funcionais, Processos-Seletivos, Bancas_Examinadoras-Comissões-Pequisas, Conhecimento-Humano, Ministra-Disciplinas, Disciplinas, Ocupação-Salas, Salas, Utilização-Recurso-Sala, Recursos, Utilização-Recurso-Laboratório, Laboratórios, Utiliza –Laboratório.
- Pedir confirmação de alteração de Docentes, Senhas, Lotação, Departamentos, Currículos,Trabalhos-Congressos, Publicações-Docentes, Capacitações-Docentes, Portarias, Progressões-Funcionais, Incentivos-Funcionais, Processos-Seletivos, Bancas_Examinadoras-Comissões-Pequisas, Conhecimento-Humano, Ministra-Disciplinas, Disciplinas, Ocupação-Salas, Salas, Utilização-Recurso-Sala, Recursos, Utilização-Recurso-Laboratório, Laboratórios, Utiliza –Laboratório.
- Consultar Docentes, Senhas, Lotação, Departamentos, Currículos,Trabalhos-Congressos, Publicações-Docentes, Capacitações-Docentes, Portarias, Progressões-Funcionais, Incentivos-Funcionais, Processos-Seletivos, Bancas_Examinadoras-Comissões-Pequisas, Conhecimento-Humano, Ministra-Disciplinas, Disciplinas, Ocupação-Salas, Salas, Utilização-Recurso-Sala, Recursos, Utilização-Recurso-Laboratório, Laboratórios, Utiliza –Laboratório.
- Sair do Sistema
- Limpar Campos da Tela

3.21. CARTÕES DAS CLASSES DE INTERFACE

Classe: Tela de Docentes	
Descrição: Classe de interface com o usuário utilizada para gerenciar os Docentes. Este gerenciamento inclui Cadastro, Alteração, Exclusão e Consulta.	
Responsabilidades identificadas: • Exibir Tela de Docentes • Fechar Tela de Docentes • Cadastrar Docentes • Alterar Docentes • Excluir Docentes • Consultar Docentes • Sair do Sistema • Limpar Campos da Tela	**Colaborações:** Docente

Classe: Tela de Senha	
Descrição: Classe de interface com o usuário utilizada para gerenciar as Senhas. Este gerenciamento inclui Cadastro, Alteração, Exclusão e Consulta.	
Responsabilidades identificadas: • Exibir Tela de Senhas • Fechar Tela de Senhas • Cadastrar Senhas • Alterar Senhas • Excluir Senhas • Consultar Senhas • Sair do Sistema • Limpar Campos da Tela	**Colaborações:** Senha

3.22. LISTA DE RESPONSABILIDADES DAS CLASSES DE PERSISTÊNCIA

Usando as responsabilidades da classe de aplicação como parâmetro obtivemos a seguinte lista de responsabilidades para as classe de persistência.:

- Consultar Docentes, Senhas, Lotação, Departamentos, Currículos,Trabalhos-Congressos, Publicações-Docentes, Capacitações-Docentes, Portarias, Progressões-Funcionais, Incentivos-Funcionais, Processos-Seletivos, Bancas_Examinadoras-Comissões-Pequisas, Conhecimento-Humano, Ministra-Disciplinas, Disciplinas, Ocupação-Salas, Salas, Utilização-Recurso-Sala, Recursos, Utilização-Recurso-Laboratório, Laboratórios, Utiliza –Laboratório.
- Gravar Docentes, Senhas, Lotação, Departamentos, Currículos,Trabalhos-Congressos, Publicações-Docentes, Capacitações-Docentes, Portarias, Progressões-Funcionais, Incentivos-Funcionais, Processos-Seletivos, Bancas_Examinadoras-Comissões-Pequisas, Conhecimento-Humano, Ministra-Disciplinas, Disciplinas, Ocupação-Salas, Salas, Utilização-Recurso-Sala, Recursos, Utilização-Recurso-Laboratório, Laboratórios, Utiliza –Laboratório.
- Excluir Docentes, Senhas, Lotação, Departamentos, Currículos,Trabalhos-Congressos, Publicações-Docentes, Capacitações-Docentes, Portarias, Progressões-Funcionais, Incentivos-Funcionais, Processos-Seletivos, Bancas_Examinadoras-Comissões-Pequisas, Conhecimento-Humano, Ministra-Disciplinas, Disciplinas, Ocupação-Salas, Salas, Utilização-Recurso-Sala, Recursos, Utilização-Recurso-Laboratório, Laboratórios, Utiliza –Laboratório.
- Alterar Docentes, Senhas, Lotação, Departamentos, Currículos,Trabalhos-Congressos, Publicações-Docentes, Capacitações-Docentes, Portarias, Progressões-Funcionais, Incentivos-Funcionais, Processos-Seletivos, Bancas_Examinadoras-Comissões-Pequisas, Conhecimento-Humano, Ministra-Disciplinas, Disciplinas, Ocupação-Salas, Salas, Utilização-Recurso-Sala, Recursos, Utilização-Recurso-Laboratório, Laboratórios, Utiliza –Laboratório.

3.23. CARTÕES DAS CLASSES DE PERSISTÊNCIA

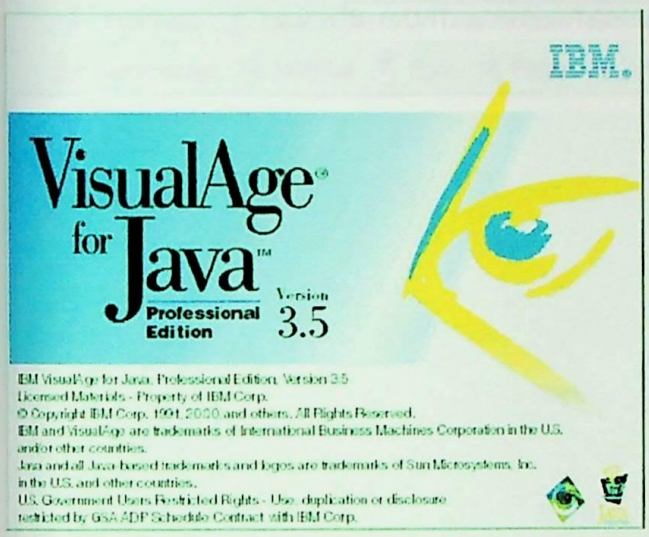
Classe: DocentePersistência	
Descrição: Classe responsável pelo armazenamento dos dados dos docentes.	
Responsabilidades identificadas: • Consultar docentes • Gravar docente • Excluir docente • Alterar docente • Retornar Dados dos Docentes	**Colaborações:** Nenhuma

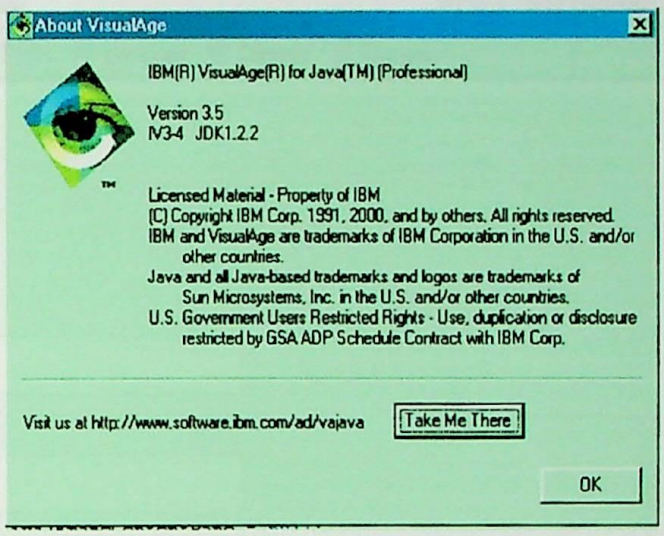
Classe: SenhaPersistência	
Descrição: Classe responsável pelo armazenamento dos dados de Senhas.	
Responsabilidades identificadas: • Consultar senhas • Gravar senhas • Alterar senha • Retornar Dados de Senha •	**Colaborações:** Nenhuma

3.24. MANUAL PRELIMINAR DO USUÁRIO

Idealmente, o protótipo (Manual Preliminar) serve como um mecanismo para identificar os requisitos de software. A construção de protótipo, possibilita que programas de trabalho sejam gerados rapidamente. [Brooks75, p.116] recomenda cuidados com o protótipo. O protótipo não é o Sistema de Informação.

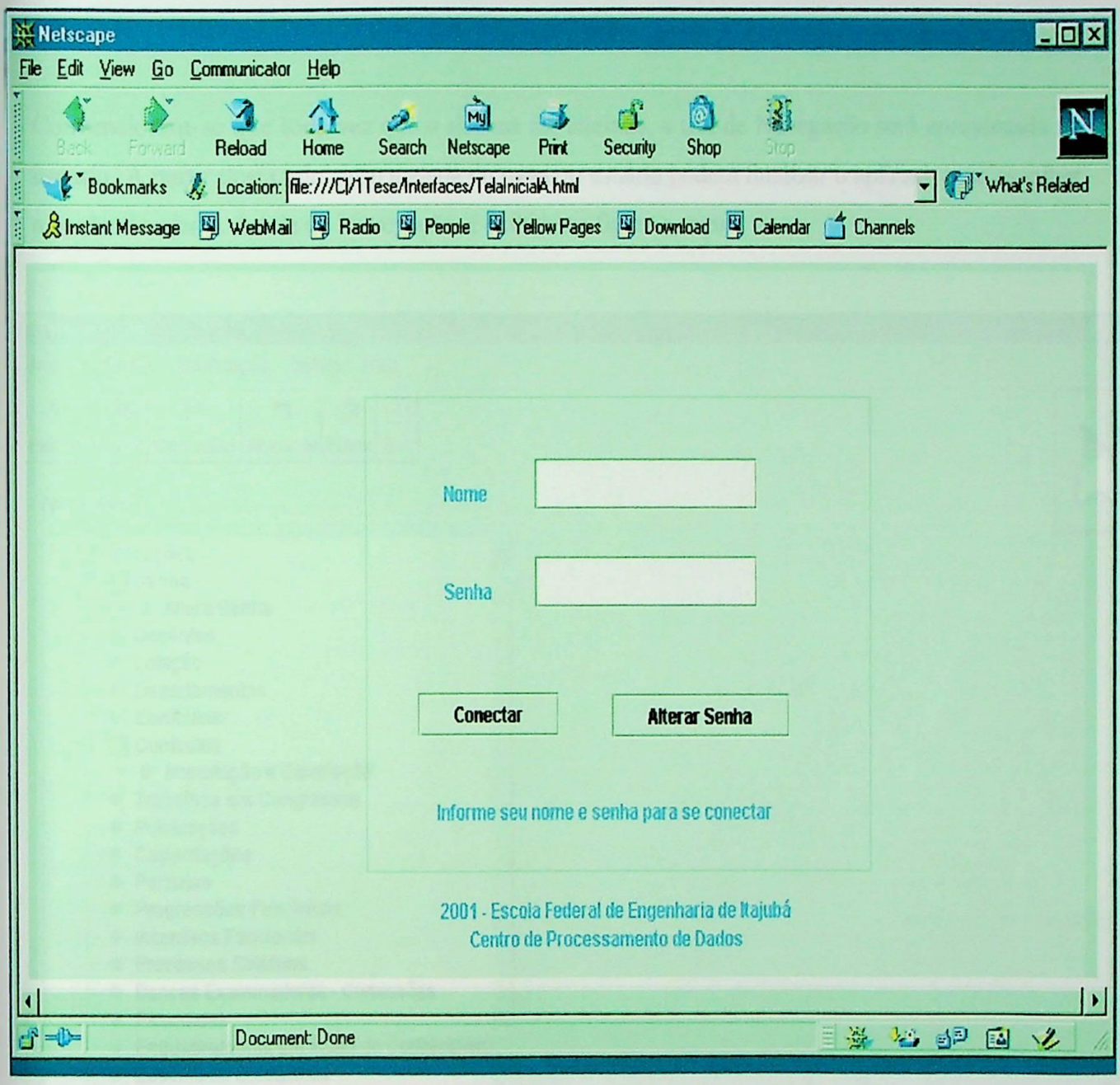
O manual preliminar do usuário, apresenta as interfaces com que os usuários irão interagir para utilizar o Sistema de Informação. Para a criação das interfaces foi usada a ferramenta VisualAge for Java Professional Edition version 3.5, como mostra as figuras abaixo.





1. Módulo Inicial - Tela Inicial

Esta tela será o primeiro contato do usuário com o Sistema. Para que seja liberado o acesso às demais telas do sistema, é necessário que o usuário seja autenticado.

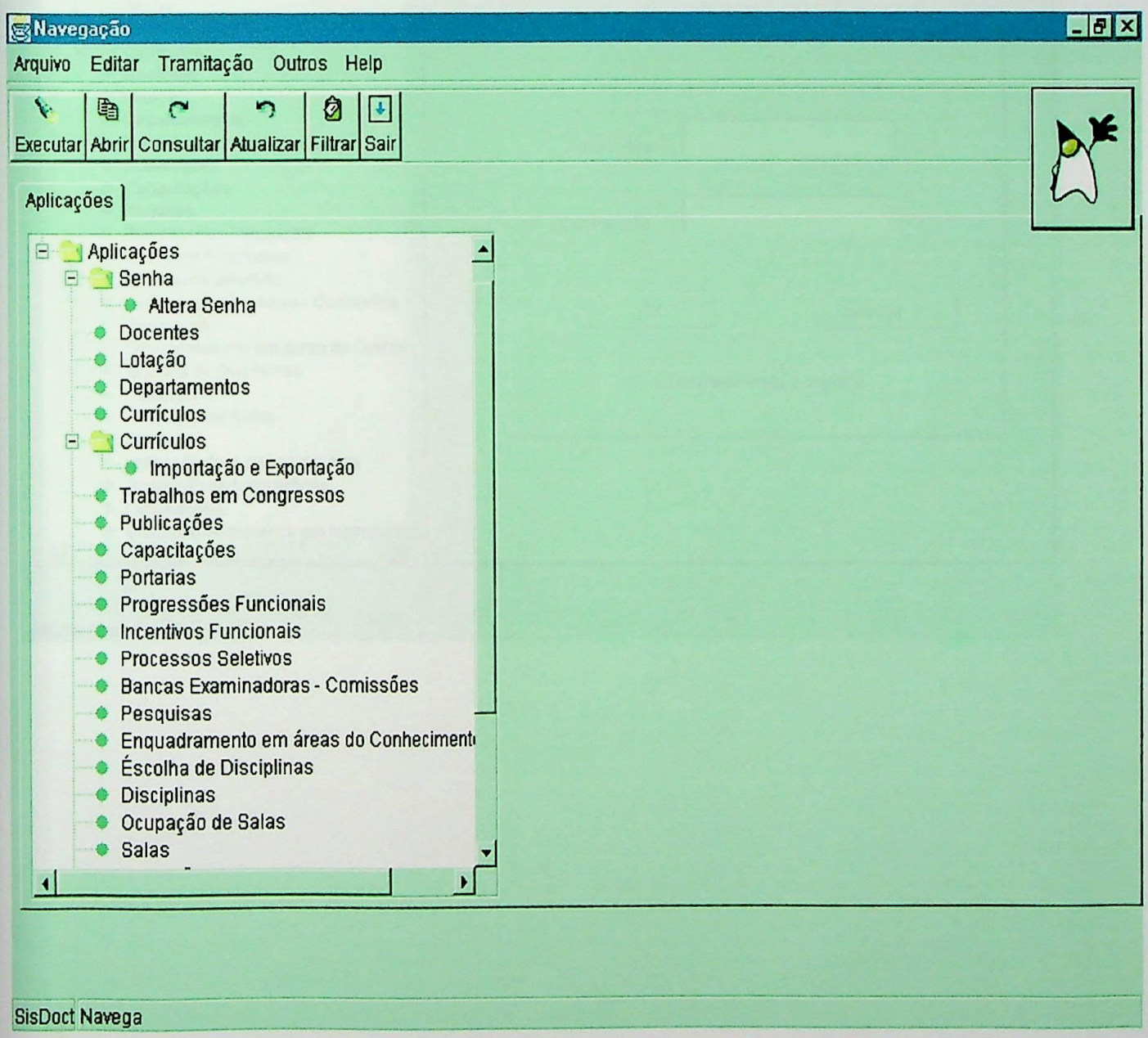


2. Módulo Navegação (Tela Inicial)

Tela principal do Sistema. Todas as aplicações do Sistema serão encontradas na forma de estrutura de diretórios. Para visualizá-las, basta acionar via mouse, o botão do sinal de (+), colocado ao lado das pastas que elas se abrirão. Todas as pastas levam um nome indicativo de aplicação, mas não executam nada, serão utilizadas somente para organização do sistema.

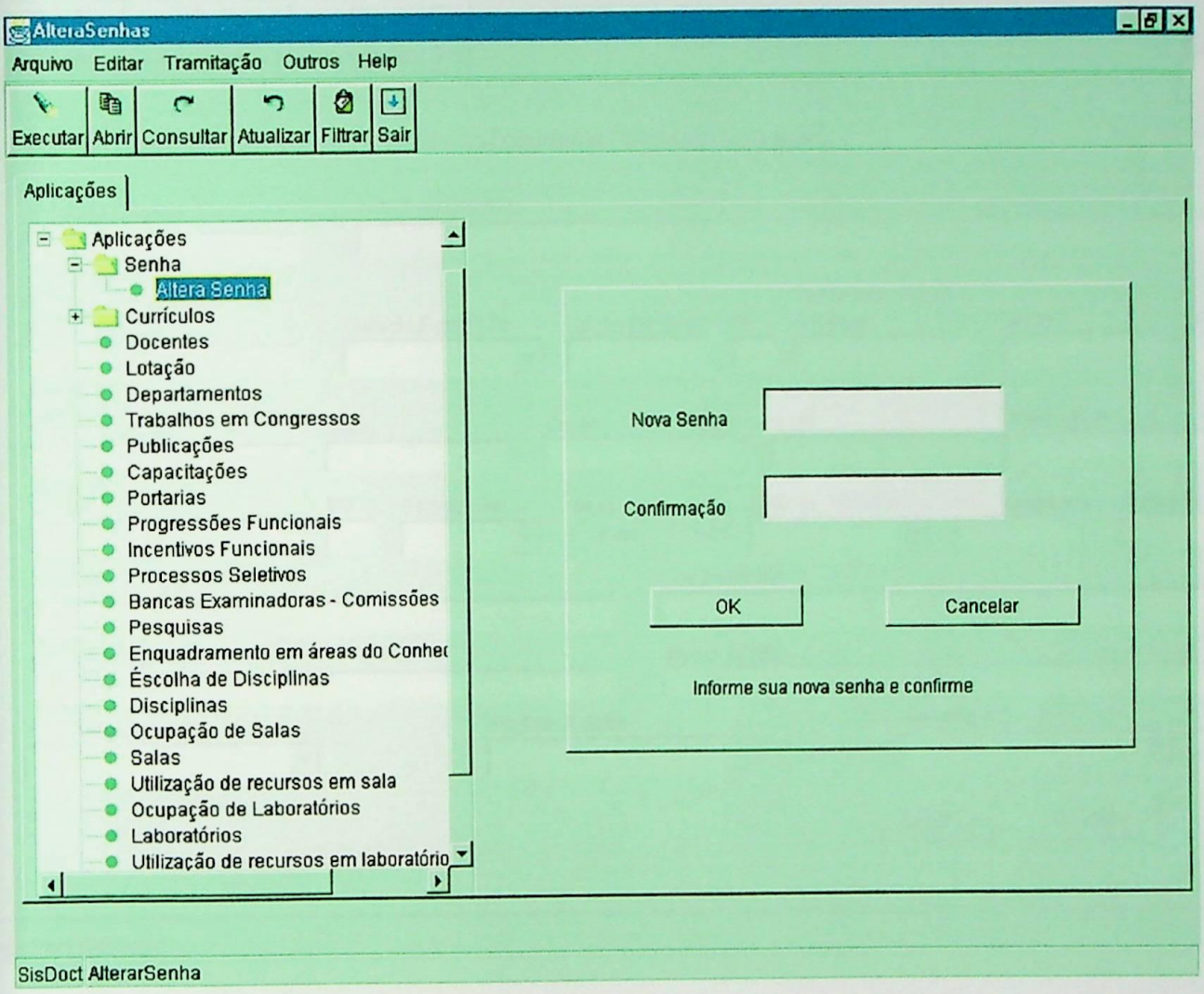
Dizemos que as pastas são itens de menu. Dentro de uma pasta podemos ter: outras pastas, aplicações, relatórios, Template ou uma aplicação Web.

Convencionou-se que toda vez que o sistema for iniciado, a tela de Navegação será apresentada ao usuário. A partir desta tela, com a ajuda do menu, o usuário poderá finalizar o aplicativo ou escolher a opção desejada. A tela de Navegação é exibida na figura a seguir.



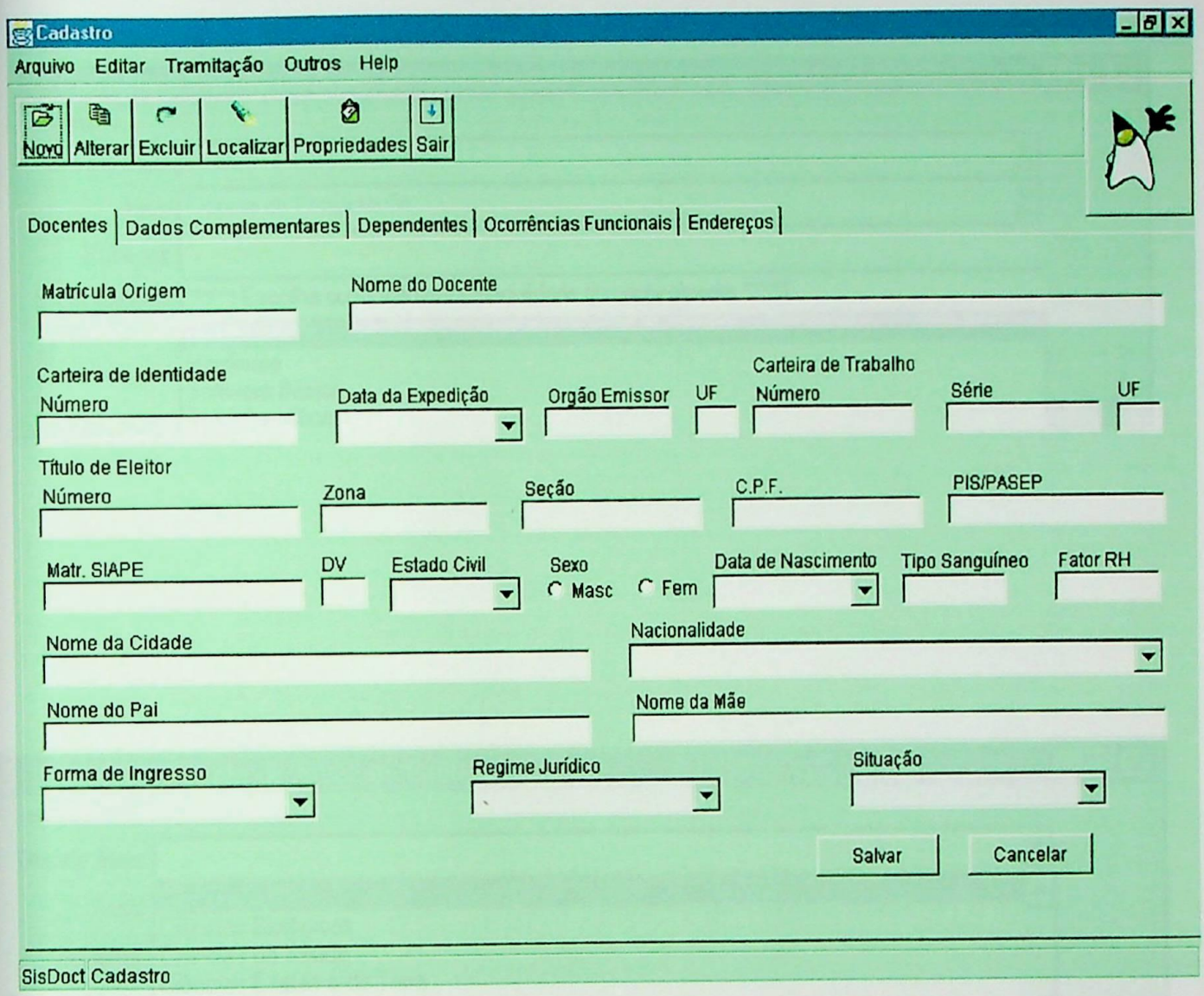
3. Módulo Alteração de Senha

Tela de alteração de senha, acionada através da tela de navegação. Esta tela permite que o usuário troque sua senha.

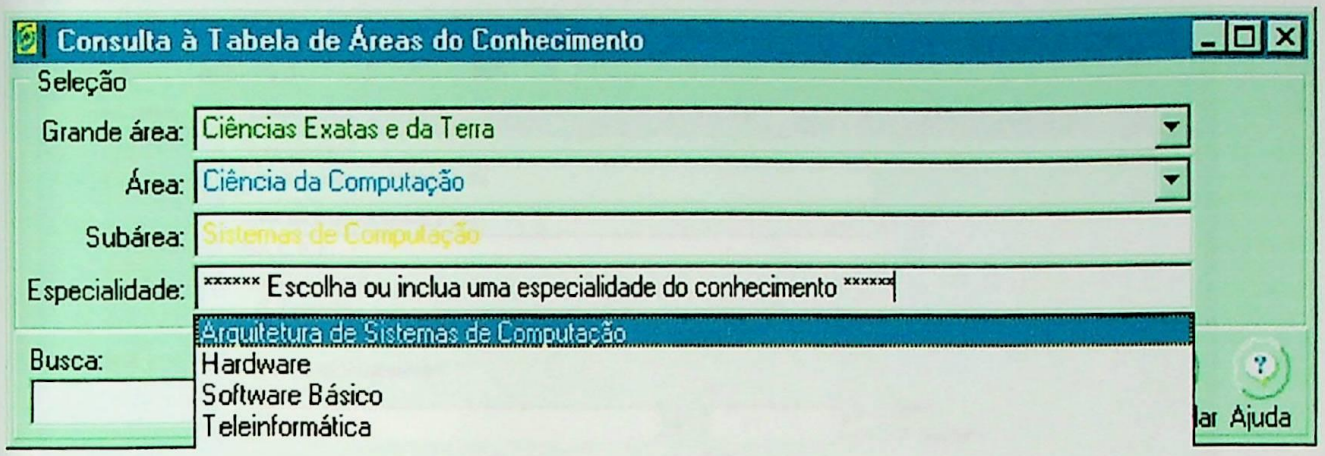


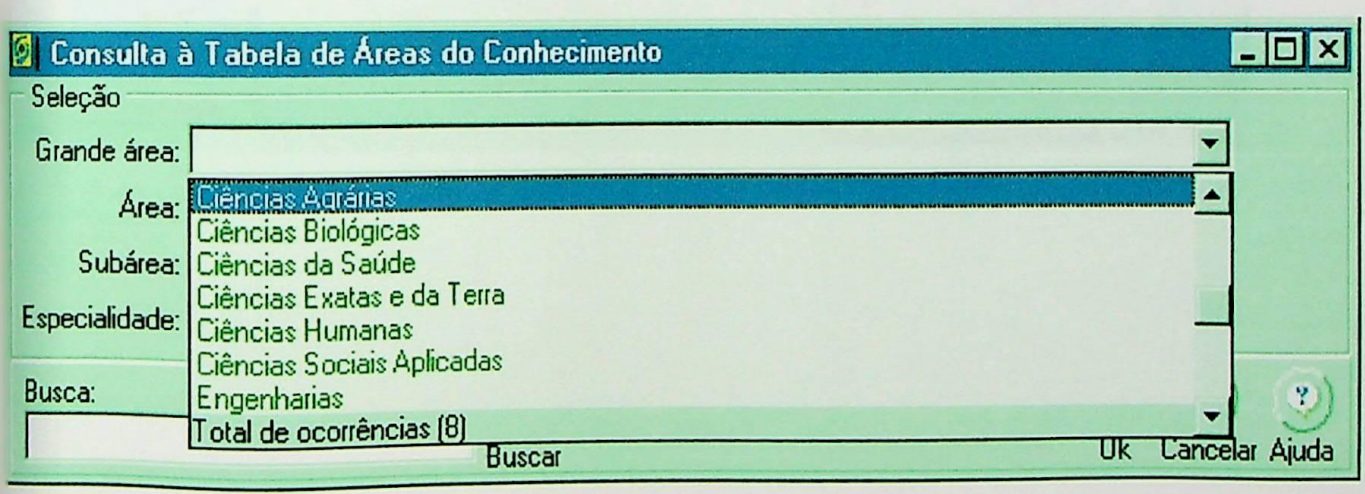
4. Módulo Docentes – Cadastro – Modificações - Consulta

Tela de dados de cadastro dos Docentes..

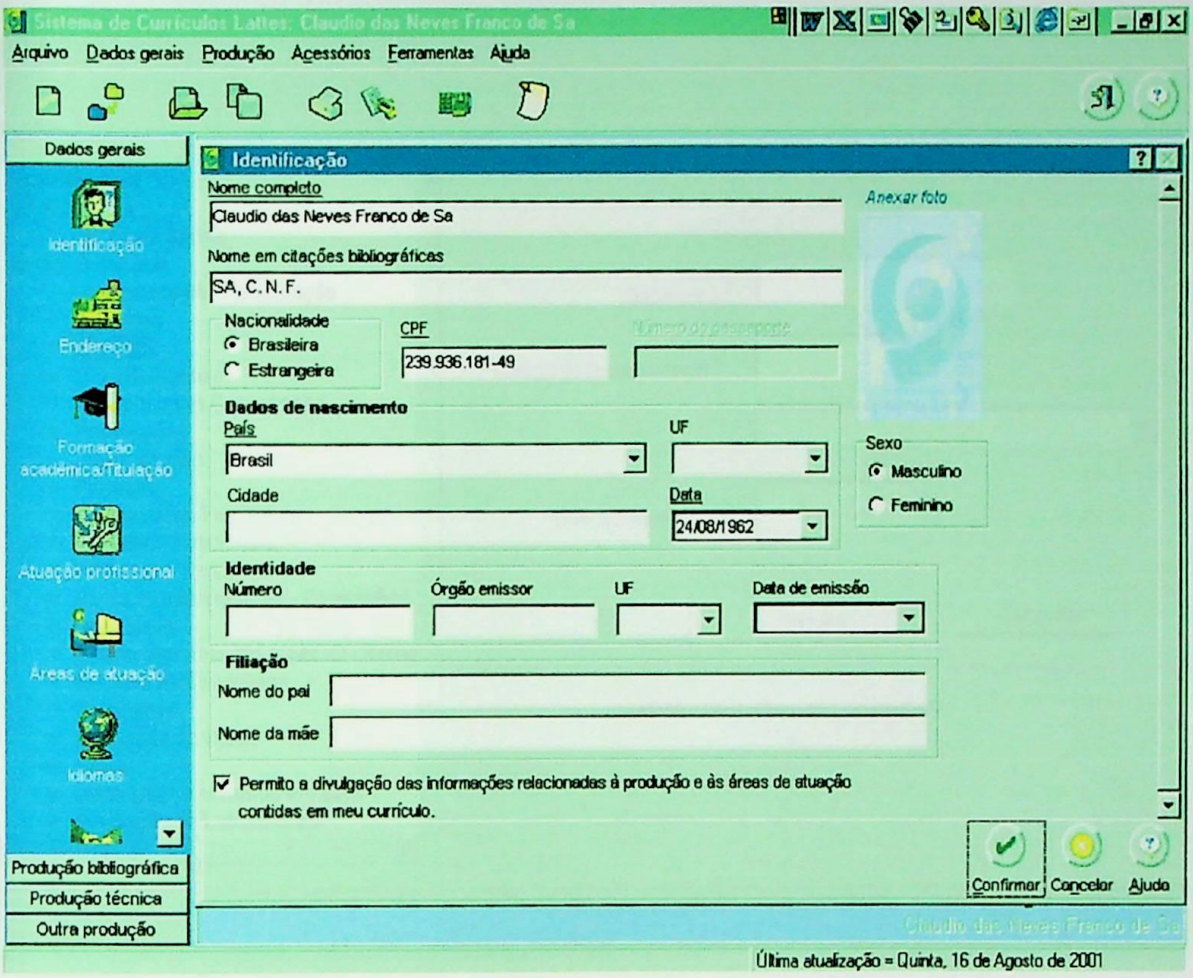


5. Módulo Áreas de Conhecimentos – Cadastro – Modificações - Consulta

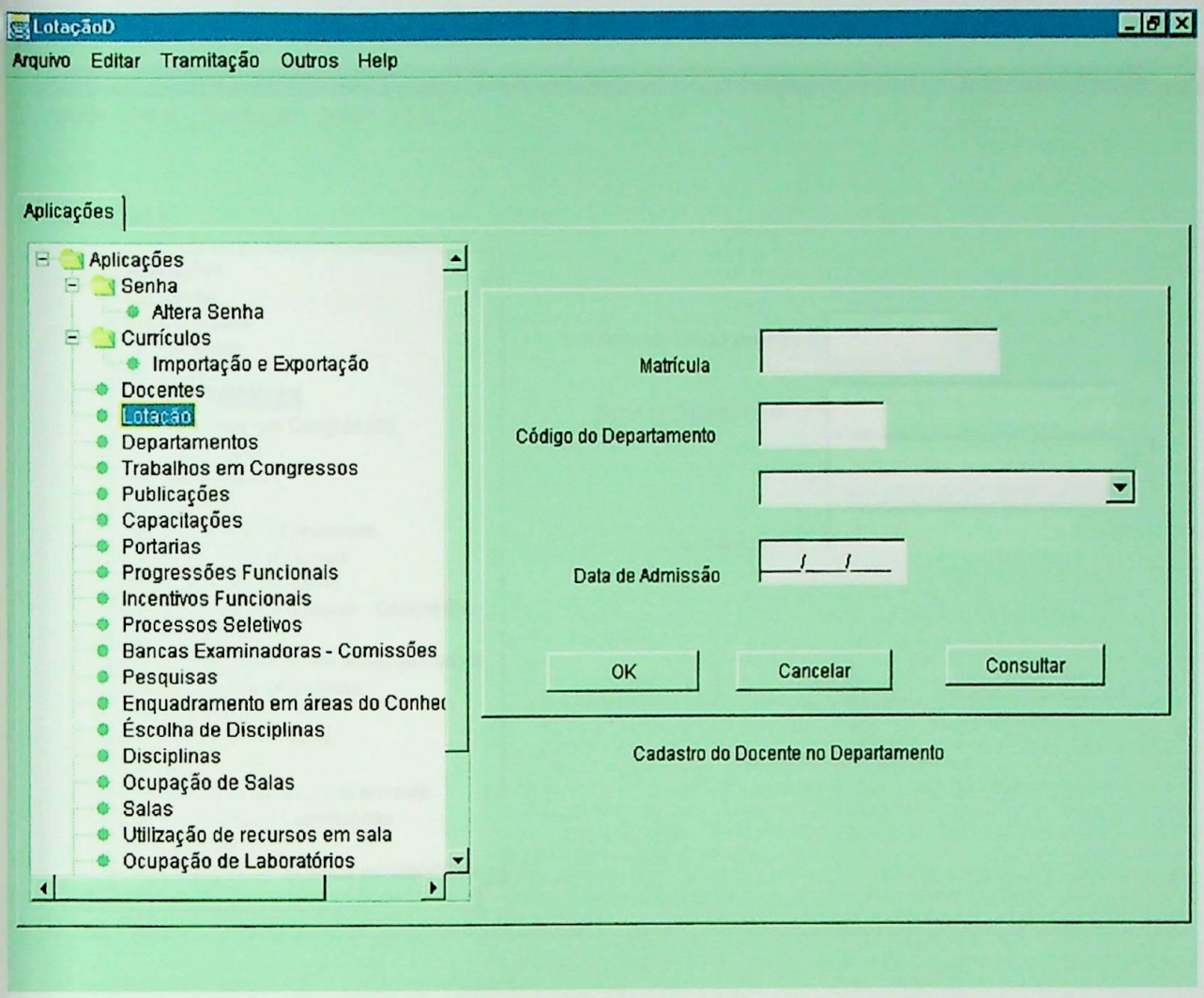




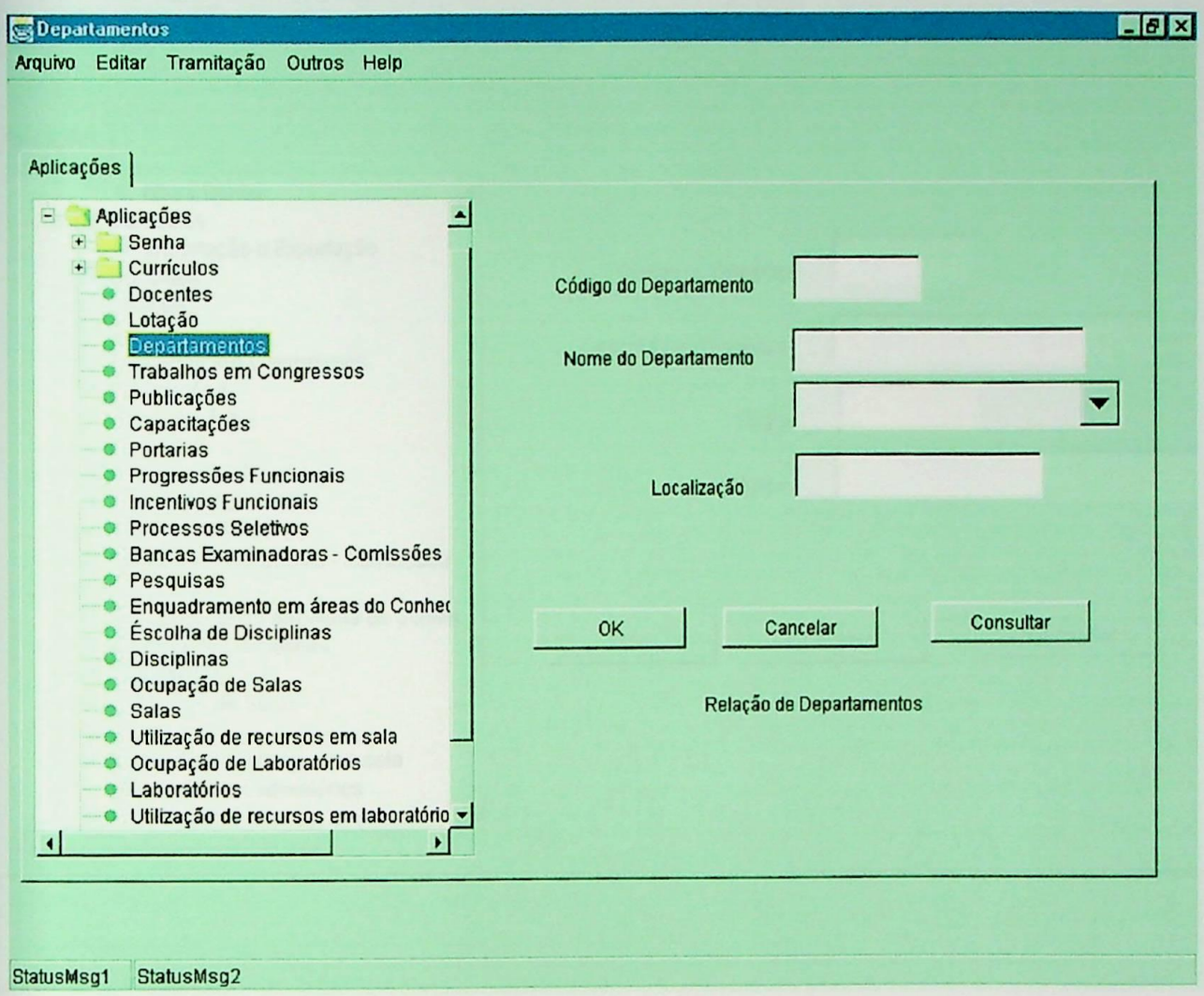
6. Módulo Currículo Lattes – Cadastro – Modificações – Consulta – Importação e Exportação



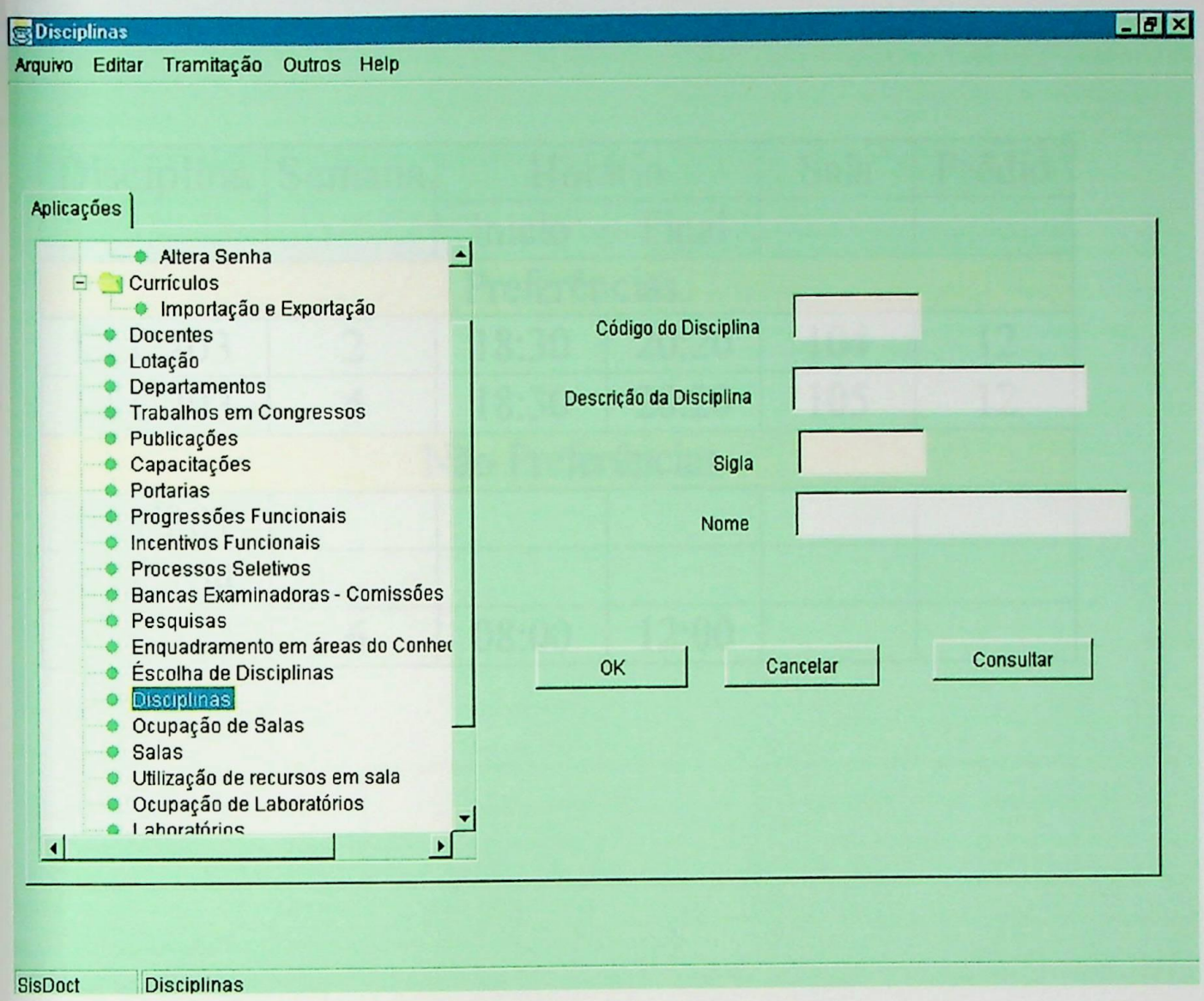
7. Módulo Lotação – Cadastro – Modificações – Consulta.



8. Módulo Departamento – Cadastro – Modificações – Consulta.



9. Módulo Disciplinas – Cadastro – Modificações – Consulta.

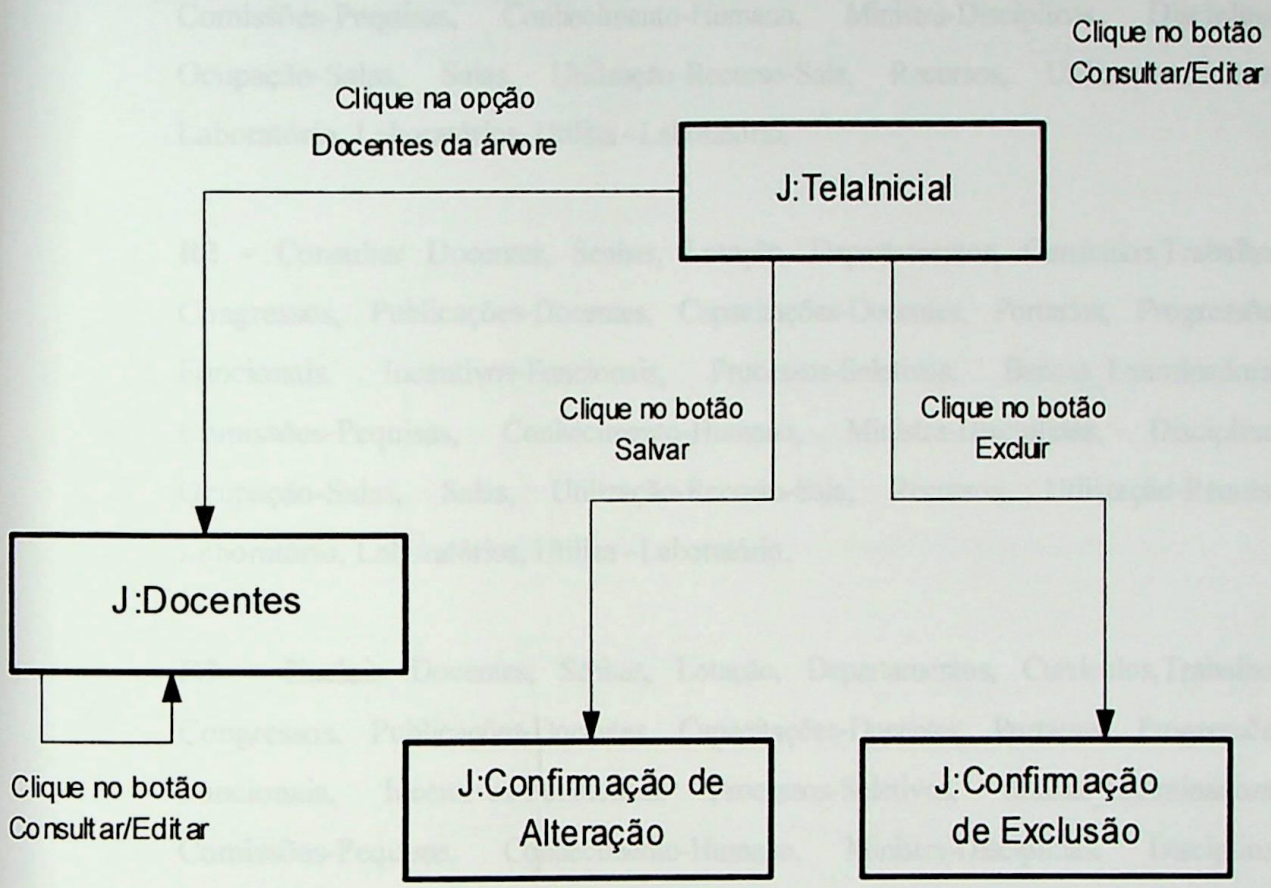


10. Modelo de escala de horário do professor.

Disciplina	Semana	Horário		Sala	Prédio
		Inicio	Final		
Preferências					
ELC203	2	18:30	20:20	104	12
ELC203	4	18:30	20:20	105	12
Não Preferências					
ELC201					
ELC204					
	6	08:00	12:00		

3.25. DIAGRAMA DE FLUXO DE INTERFACES

Esta ferramenta é usada como mais um recurso de ajuda ao usuário. Por ser gráfica, permite uma melhor visualização e acompanhamento do fluxo de execução das opções disponíveis.



3.26. FASE EXPLORATÓRIO COM CENÁRIOS (FEC)

3.26.1. RESPONSABILIDADES POR ORDEM DE PRIORIDADE DAS CLASSES DE APLICAÇÃO

R1 - Cadastrar Docentes, Senhas, Lotação, Departamentos, Currículos,Trabalhos-Congressos, Publicações-Docentes, Capacitações-Docentes, Portarias, Progressões-Funcionais, Incentivos-Funcionais, Processos-Seletivos, Bancas_Examinadoras-Comissões-Pequisas, Conhecimento-Humano, Ministra-Disciplinas, Disciplinas, Ocupação-Salas, Salas, Utilização-Recurso-Sala, Recursos, Utilização-Recurso-Laboratório, Laboratórios, Utiliza –Laboratório.

R2 - Consultar Docentes, Senhas, Lotação, Departamentos, Currículos,Trabalhos-Congressos, Publicações-Docentes, Capacitações-Docentes, Portarias, Progressões-Funcionais, Incentivos-Funcionais, Processos-Seletivos, Bancas_Examinadoras-Comissões-Pequisas, Conhecimento-Humano, Ministra-Disciplinas, Disciplinas, Ocupação-Salas, Salas, Utilização-Recurso-Sala, Recursos, Utilização-Recurso-Laboratório, Laboratórios, Utiliza –Laboratório.

R3 - Excluir Docentes, Senhas, Lotação, Departamentos, Currículos,Trabalhos-Congressos, Publicações-Docentes, Capacitações-Docentes, Portarias, Progressões-Funcionais, Incentivos-Funcionais, Processos-Seletivos, Bancas_Examinadoras-Comissões-Pequisas, Conhecimento-Humano, Ministra-Disciplinas, Disciplinas, Ocupação-Salas, Salas, Utilização-Recurso-Sala, Recursos, Utilização-Recurso-Laboratório, Laboratórios, Utiliza –Laboratório.

R4 - Alterar Docentes, Senhas, Lotação, Departamentos, Currículos,Trabalhos-Congressos, Publicações-Docentes, Capacitações-Docentes, Portarias, Progressões-Funcionais, Incentivos-Funcionais, Processos-Seletivos, Bancas_Examinadoras-Comissões-Pequisas, Conhecimento-Humano, Ministra-Disciplinas, Disciplinas, Ocupação-Salas, Salas, Utilização-Recurso-Sala, Recursos, Utilização-Recurso-Laboratório, Laboratórios, Utiliza –Laboratório.

3.26.2. CLASS-RESPONSIBILITY-COLLABORATION (CRC) DESCRIÇÃO DAS RESPONSABILIDADES E COLABORAÇÕES

Responsabilidade	R1.1 - Cadastrar docentes
Classe	Docente
Colaboração	
Processamento Lógico	Administrador preenche campos e solicita cadastro. Sistema verifica o se foi inserido algo nos campos. Em caso positivo, docente é cadastrado, senão sistema não habilita o botão salvar.

Responsabilidade	R1.2 – Cadastrar senhas
Classe	Senha
Colaboração	
Processamento Lógico	Administrador preenche campos e solicita cadastro. Sistema verifica o campo nome e verifica existência do usuário. Em caso positivo, solicita senha e redigitação da senha. Verificado possíveis dígitos inválidos, se tudo ok, senha é cadastrada, se não um aviso de redigitação de senha será exibido.

Responsabilidade	R2.1 - Consultar Docentes
Classe	Docentes
Colaboração	
Processamento Lógico	Usuário preenche campo(s) e solicita consulta. Sistema verifica o(s) campo(s) para ver se o(s) mesmo(s) existe(m). Em caso positivo docente é exibido, senão nada é retornado pelo sistema.

Responsabilidade	R3.1 - Excluir Docentes
Classe	Docente
Colaboração	
Processamento Lógico	Administrador identifica docente a ser eliminado e solicita exclusão. Sistema emite mensagem solicitando confirmação. Se confirmada a exclusão o sistema a executa, senão cancela a transação.

Responsabilidade	R4.1 - Alterar Docentes
Classe	Docente
Colaboração	
Processamento Lógico	Administrador identifica o docente a ser modificado. Sistema mostra informações cadastradas do docente, libera campos de dados para alteração e

	botão de confirmação de alteração. Se confirmada a alteração o sistema a executa, senão cancela a transação.

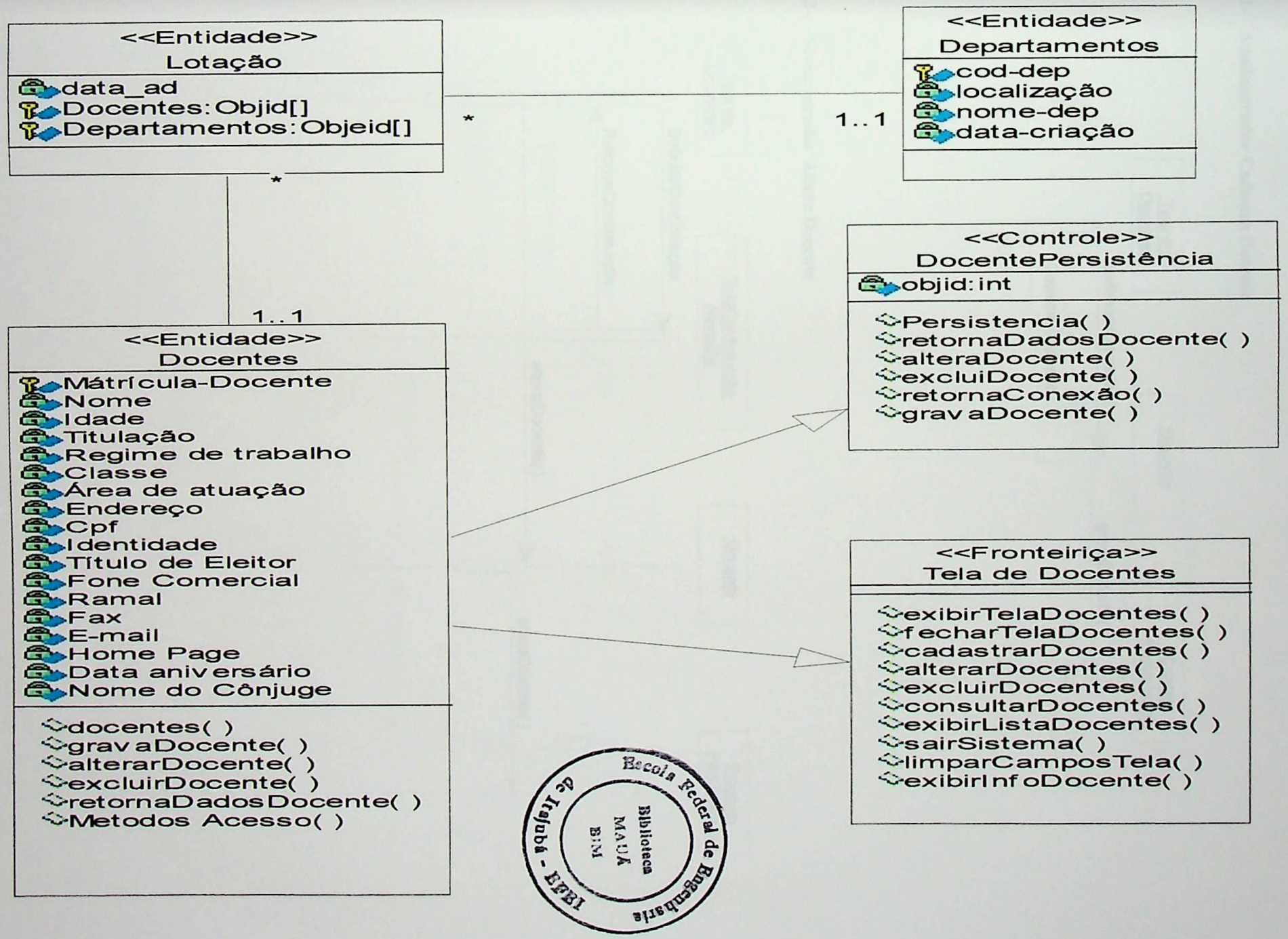
3.26.3. RELAÇÕES DE DEPENDÊNCIA DAS RESPONSABILIDADES ENTRE AS CLASSES DE APLICAÇÃO

Devido à natureza independente das classes docentes, e pelas responsabilidades levantadas, não existe relação de dependência entre as responsabilidades das classes de aplicação.

3.27. DIAGRAMA DE CLASSES (Fase de Consolidação e Relacionamento – Visão de projeto)

O diagrama de classes demonstra a estrutura estática das classes de um sistema onde estas representam as "coisas" que são gerenciadas pela aplicação modelada. Classes podem se relacionar com outras através de diversas maneiras: associação (conectadas entre si), dependência (uma classe depende ou usa outra classe), especialização (uma classe é uma especialização de outra classe), ou em pacotes (classes agrupadas por características similares).

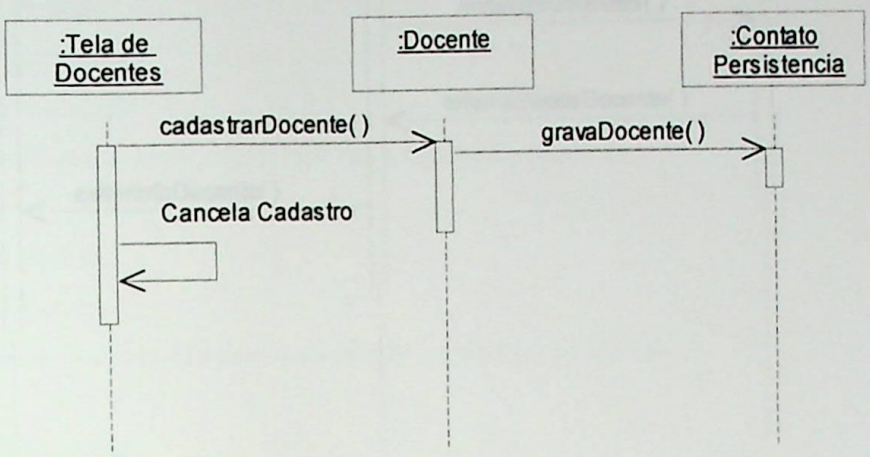
Um *diagrama de classes* mostra um conjunto de classes, interfaces e colaborações e seus relacionamentos.



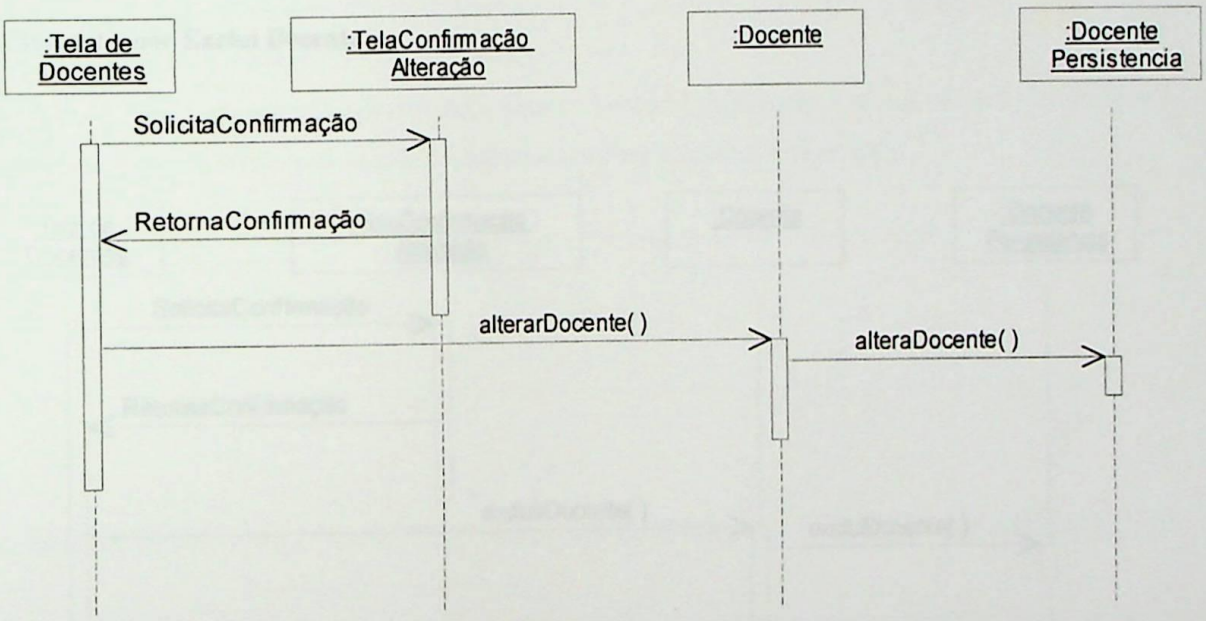
3.28. DIAGRAMA DE SEQÚÊNCIA

Os diagramas de seqüência serão listados segundo os Use Cases. Para cada Use Case será feito um diagrama de seqüência.

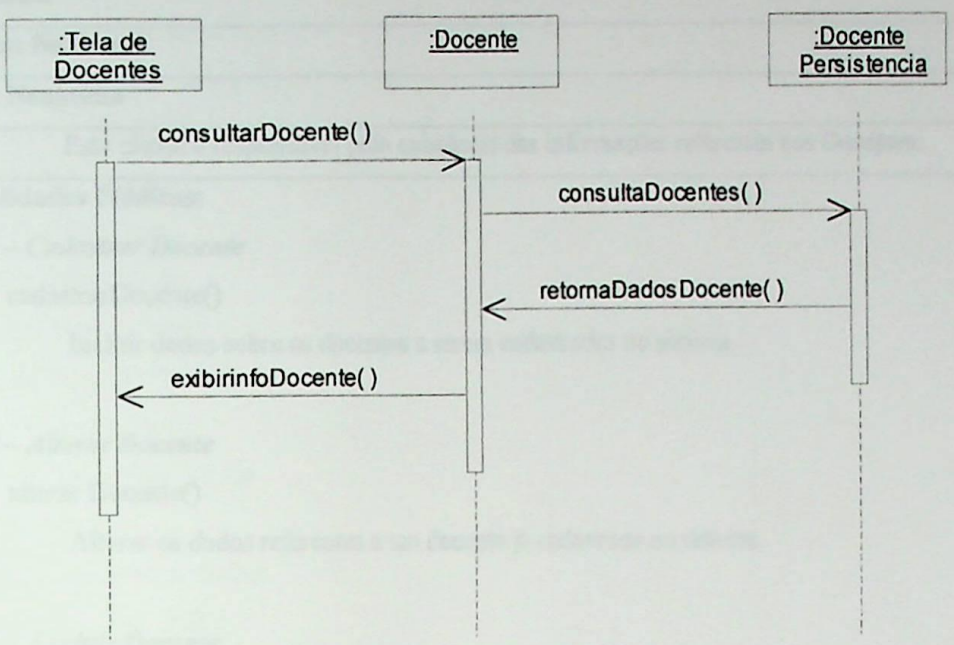
1- Administrador Cadastra Docente



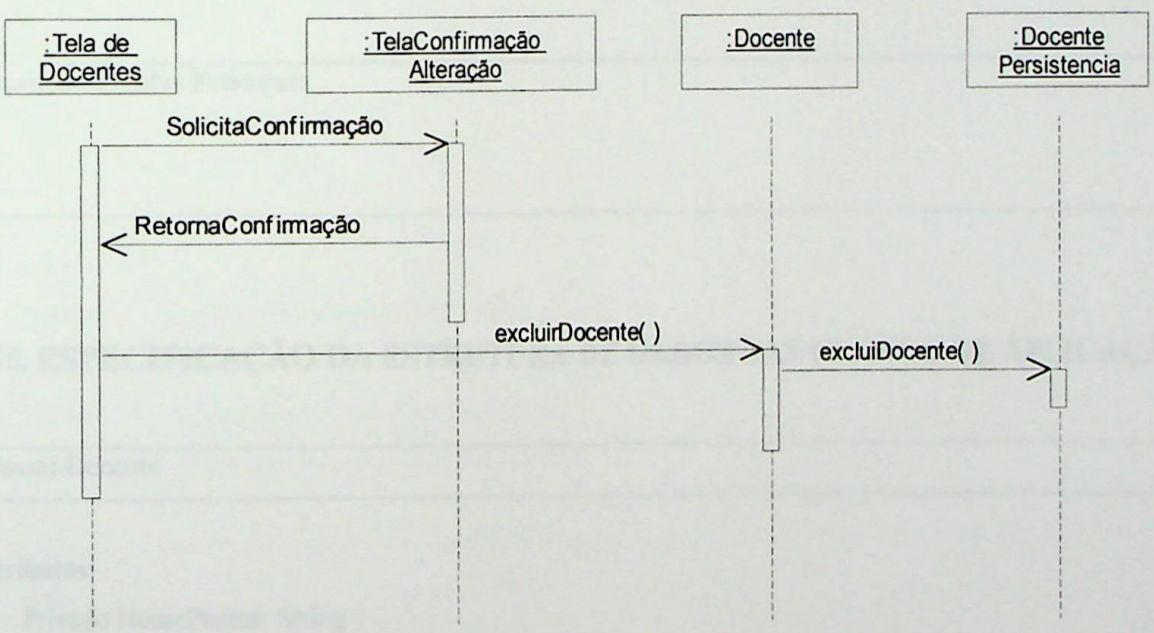
2- Administrador Altera Docente



3- Usuário Consulta Docente



4- Administrador Exclui Docente



3.29. CARTÕES FINAIS

Classe: Docente	(Concreta)
Superclasses: Nenhuma	
Subclasses: Nenhuma	
Descrição: Esta classe é responsável pelo existência das informações referentes aos Docentes.	

Responsabilidades Públicas:

R1 – Cadastrar Docente

cadastrarDocente()

Incluir dados sobre os docentes a serem cadastrados no sistema.

R2 – Alterar Docente

alterar Docente()

Alterar os dados referentes a um docente já cadastrado no sistema.

R3 – Excluir Docente

excluir Docente()

Excluir todos os dados referente a um docente já cadastrado.

R4 – Consultar Docente

Consultar Docente()

Executa consulta parametrizada sobre os docentes já cadastrados e retorna os resultados ao usuário na forma de uma lista de docentes que atendem às condições da consulta.

Responsabilidades Privadas:

3.30. ESPECIFICAÇÃO DA ESTRUTURA DE DADOS DAS CLASSES DE APLICAÇÃO

Classe: Docente

Atributos:

Privado NomePessoa String
Privado IDDocente String
Privado DataAniversario String
Privado Conjugue String
// Endereço Comercial
Privado ECCargonaEmpresa String
Privado ECNomeEmpresa String

Privado ECLogradouro String
Privado ECNúmero String
Privado ECComplemento String
Privado ECBairroDistrito String
Privado ECMunicípio String
Privado ECEstado String
Privado ECPais String
Privado ECCEP String
Privado ECFone String
Privado ECRamal String
Privado ECFax String
Privado ECEmail String
Privado ECHomePage String
// Endereço Residencial
Privado ERLogradouro String
Privado ERNúmero String
Privado ERComplemento String
Privado ERBairroDistrito String
Privado ERMunicípio String
Privado EREstado String
Privado ERPais String
Privado ERCEP String
Privado ERFone String
Privado ERFax String
Privado EREmail String
Privado ERHomePage String
Privado ERCelular String

Métodos:

Público CadastrarDocente (string Vet[]);
Público AlterarDocente(string Vet[]);
Público ExcluirDocente(String IDDocente);
Público ConsultarDocente(String IDDocente);

3.31. ESPECIFICAÇÃO DOS MÉTODOS

Classe: Docente — Concreta

Superclasse: Nenhuma

Métodos:

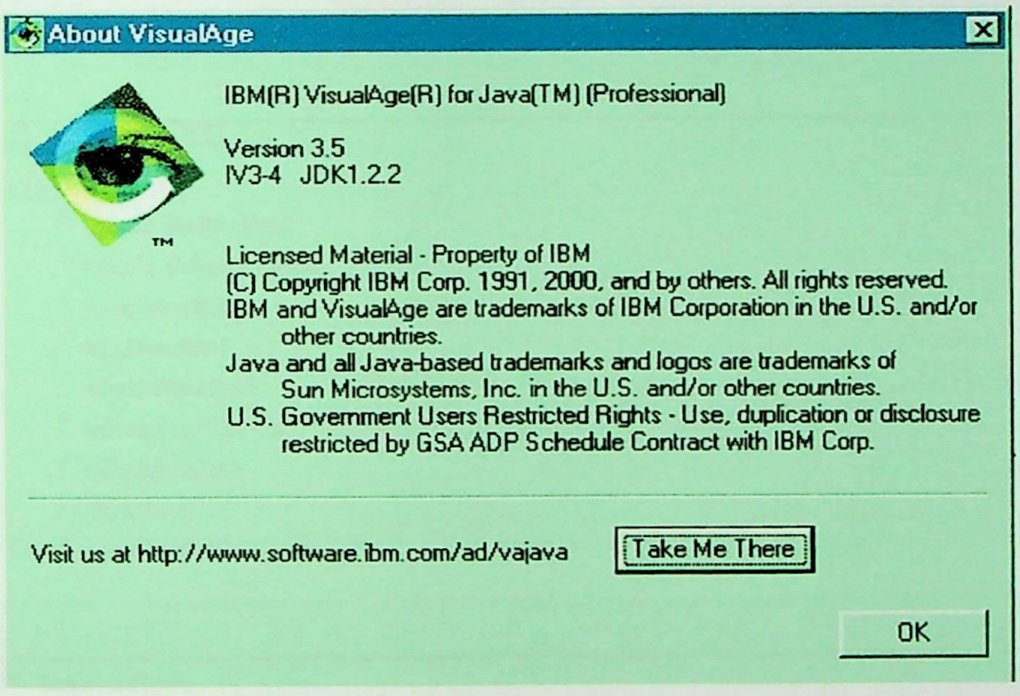
```
publico CadastrarDocente(String Vet[20])
inicio
   Enquanto (inteiro i==0; i<=20;i++) inicio
         //utilizar todos os métodos de acesso gravar... (vet[i]) para todos os campos
         Salvar os campos na base de dados
         Ordenar base de dados; fim
fim

 publico ExcluirDocente(Inteiro IDDocente);
      inicio
     Se (IDDocente == -1) então erro("Registro não encontrado na Base de Dados")
     Senão inicio
            Se (IDDocente == obterIDDocente()) inicio
           Remover registro "IDDocente" da base de dados;
                 Ordenar base de dados; fim
fim

publico AlterarDocente(inteiro IDDocente);
inicio
   Se (IDDocente == -1) então erro("Registro não encontrado na Base de Dados")
     Senão inicio
            Se (IDDocente == obterIDDocente()) inicio
               Para (inteiro i =1; i<= 20, i++) inicio
               obter...() os campos através dos métodos de acesso
               //utilizar todos os métodos de acesso gravar... (campo) para todos os campos
               Salvar os campos na base de dados
               Ordenar base de dados; fim

      fim
   fim
fim

publico ConsultarDocente(inteiro IDDocente);
inicio
```

```
    Se (IDDocente == -1) então erro("Registro não encontrado na Base de Dados")
      Senão inicio
            Se (IDDocente == obterIDDocente()) inicio
               Para (inteiro i =1; i<= 20, i++) inicio
               obter...() os campos através dos métodos de acesso
       fim
     fim
  fim

//Métodos de Acesso

privado obterIDDocente() inicio
  retorna (IDDocente);
fim
privado gravarIDDocente(IDDocente) inicio
  this.IDDocente = IDDocente;
fim
privado obterNomePessoa() inicio
  retorna (IDDocente);
fim
```

3.32. ESPECIFICAÇÃO DA IMPLEMENTAÇÃO (ANÁLISE DE REUSO)

O documento SISDOCT, conforme especificado, foi trabalhado com classes de interface, classes de negócio (também chamadas de classes da aplicação) e classes de persistência. A partir de agora iremos especificar os reusos encontrados em todos os tipos de classes (No sistema SISDOCT o tipo de reuso encontrado foi o direto das classes da biblioteca do java). Como trabalhamos (implementamos) em Java, especificaremos o reuso direto das bibliotecas (classes) do mesmo.

Com relação às classes de interface, como utilizamos uma ferramenta RAD (IBM VisualAge), destacamos em todas as classes o reuso direto das classes da biblioteca do java, mais precisamente classes do package AWT e SWING. Como trata-se de interface, o uso de botões, janelas, campos de edição, etc..., são comuns. Por isto o reuso de determinadas classes do Java não é estranho. Também foram reutilizadas classes originadas de pacotes da ferramenta de desenvolvimento (IBM VisualAge).



3.32.1. IMPORTAÇÃO DE PACOTES E REUSO DAS CLASSES DE INTERFACE

Pacotes:

import java.awt.*;

import javax.swing.*;

import javax.swing.tree.*;

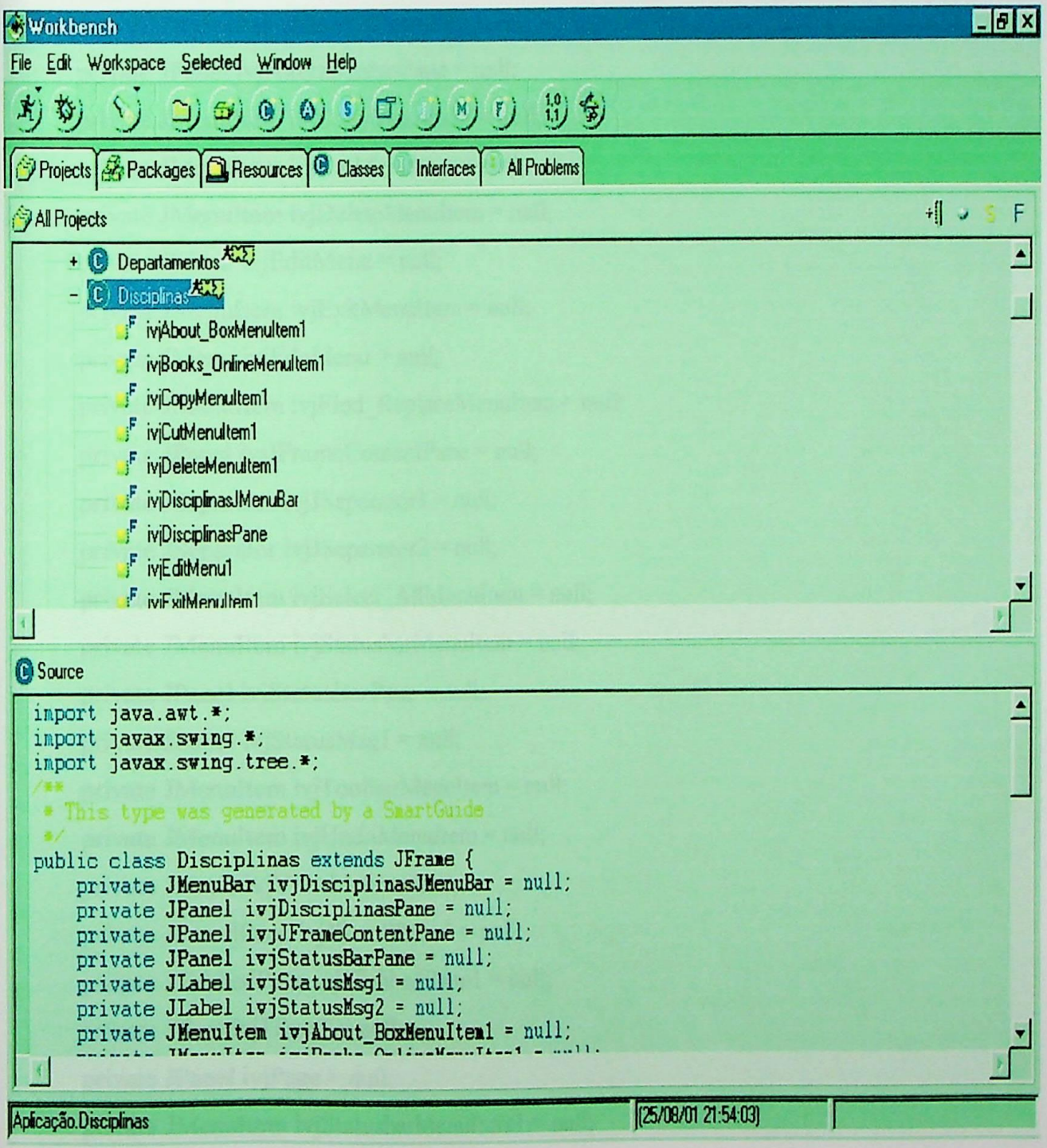
Classes:

Jpanel; JmenuItem; Jmenu; Jseparator; Jlabel;

JscrollPane; JtabbedPane; Jtree; Jbutton;

JtextField; BorderLayout; JcomboBox; JradioButton;

JtoolBar.



Classe (class) telaDocentes.

Reuso das seguintes classes do package SWING do java e os respectivos objetos criados a partir das mesmas no sistema SISDOCT:

```
import java.awt.*;
import javax.swing.*;
import javax.swing.tree.*;
/**
 * This type was generated by a SmartGuide.
 * Este programa trata da tela AlteraSenhas.
 */
public class AlteraSenhas extends JFrame {
        private JPanel ivjAlteraSenhasPane = null;
        private JMenuItem ivjCopyMenuItem = null;
        private JMenuItem ivjCutMenuItem = null;
        private JMenuItem ivjDeleteMenuItem = null;
        private JMenu ivjEditMenu = null;
        private JMenuItem ivjExitMenuItem = null;
        private JMenu ivjFileMenu = null;
        private JMenuItem ivjFind_ReplaceMenuItem = null;
        private JPanel ivjJFrameContentPane = null;
        private JSeparator ivjJSeparator1 = null;
        private JSeparator ivjJSeparator2 = null;
        private JMenuItem ivjSelect_AllMenuItem = null;
        private JMenuItem ivjStatusbarMenuItem = null;
        private JPanel ivjStatusBarPane = null;
        private JLabel ivjStatusMsg1 = null;
        private JMenuItem ivjToolbarMenuItem = null;
        private JMenuItem ivjUndoMenuItem = null;
        private JMenu ivjViewMenu = null;
        private JScrollPane ivjJScrollPane1 = null;
        private JTabbedPane ivjJTabbedPane1 = null;
        private JTree ivjJTree1 = null;
        private JPanel ivjPage = null;
        private JMenuItem ivjStatusbarMenuItem1 = null;
```

```
        private JMenuItem ivjToolbarMenuItem1 = null;
        private JMenu ivjViewMenu1 = null;
        private JMenuBar ivjAlteraSenhasJMenuBar = null;
        private JButton ivjJButton1 = null;
        private JLabel ivjJLabel2 = null;
        private JTextField ivjJTextField1 = null;
        private JTextField ivjJTextField11 = null;
        private BorderLayout ivjJFrameContentPaneBorderLayout = null;
        private BorderLayout ivjStatusBarPaneBorderLayout = null;
        private JMenuItem ivjAbout_BoxMenuItem1 = null;
        private JMenuItem ivjBooks_OnlineMenuItem1 = null;
        private JMenuItem ivjHelp_TopicsMenuItem1 = null;
        private JMenu ivjHelpMenu1 = null;
}
import java.awt.*;
import javax.swing.*;
/**
 * This type was generated by a SmartGuide.
 * Este programa trata da tela de Cadastro.
 */
public class Cadastro extends JFrame {
        private JMenuBar ivjCadastroJMenuBar = null;
        private JPanel ivjCadastroPane = null;
        private JLabel ivjStatusMsg2 = null;
        private JMenuItem ivjAbout_BoxMenuItem1 = null;
        private JMenuItem ivjDeleteMenuItem11 = null;
        private JMenu ivjEditMenu1 = null;
        private JMenu ivjEditMenu2 = null;
        private JMenu ivjFileMenu1 = null;
        private JMenuItem ivjFind_ReplaceMenuItem1 = null;
        private JMenuItem ivjHelp_TopicsMenuItem1 = null;
        private JMenu ivjHelpMenu1 = null;
        private JSeparator ivjJSeparator11 = null;
        private JSeparator ivjJSeparator21 = null;
        private JMenuItem ivjNewMenuItem1 = null;
```

```
    private JMenuItem ivjUndoMenuItem1 = null;
    private JMenuItem ivjUndoMenuItem11 = null;
    private JMenu ivjViewMenu1 = null;
    private JLabel ivjJLabel1 = null;
    private JLabel ivjJLabel11 = null;
    private JTabbedPane ivjJTabbedPane1 = null;
    private JTextField ivjJTextField1 = null;
    private JTextField ivjJTextField11 = null;
    private JPanel ivjPage = null;
    private JComboBox ivjJComboBox1 = null;
    private JLabel ivjJLabel12 = null;
    private JLabel ivjJLabel121 = null;
    private JTextField ivjJTextField12 = null;
    private JComboBox ivjJComboBox11 = null;
    private JLabel ivjJLabel121411 = null;
    private JRadioButton ivjJRadioButton1 = null;
    private JRadioButton ivjJRadioButton2 = null;
    private JTextField ivjJTextField123111 = null;
    private JTextField ivjJTextField1231111 = null;
    private JButton ivjJButton1 = null;
    private JButton ivjJButton2 = null;
    private BorderLayout ivjJFrameContentPaneBorderLayout = null;
    private BorderLayout ivjStatusBarPaneBorderLayout = null;
    private JButton ivjCopyButton = null;
    private JToolBar ivjToolBarPane = null;
    private JPanel ivjJPanel1 = null;
    private JPanel ivjJPanel4 = null;
}
```

{ INÍCIO DOS ATRIBUTOS}

java.swing.TextField objeto local => txtECCargoEmpresa

java.swing.TextField objeto local => txtECNomeEmpresa

java.swing.TextField objeto local => txtECLogradouro

java.swing.TextField objeto local => txtECNumero

java.swing.TextField objeto local => txtECComplemento

java.swing.TextField objeto local => txtECBairro

java.swing.TextField objeto local => txtECMunicipio

java.swing.TextField objeto local => txtECEstado

java.swing.TextField objeto local => txtECPais

java.swing.TextField objeto local => txtECRamal

java.swing.TextField objeto local => txtECEmail

java.swing.TextField objeto local => txtECHomePage

java.swing.TextField objeto local => txtNomeConjuge

java.swing.TextField objeto local => txtLogradouro

java.swing.TextField objeto local => txtNumero

java.swing.TextField objeto local => txtComplemento

java.swing.TextField objeto local => txtBairro

java.swing.TextField objeto local => txtMunicipio

java.swing.TextField objeto local => txtEstado

java.swing.TextField objeto local => txtPais

java.swing.TextField objeto local => txtEmailPessoal

java.swing.TextField objeto local => txtHPPessoal

Reuso das classes de Persistência e de Aplicação :

Com relação às classes de aplicação, não temos a utilização de nenhum package do Java, pois ela somente faz referência a métodos das classes de persistência, ou seja, quem faz realmente o trabalho direto com o BD são as classes de persistência. Nestas classes destacamos a utilização do package Java.Sql. Este pacote JDBC possibilita a execução de comandos SQL, comuns às classes de persistência: DocentePersistencia. Estas, como já foi dito, fazem acesso ao Banco de Dados Access da SISDOCT.

Classe (class) DocentePersistencia.

Reuso da seguinte classe do package Lang no sistema SISDOCT:

Class.forName

Reuso das seguintes classes do package Sql do java e do package Útil.Vetor. Assim os respectivos objetos criados a partir das classes destes packages no sistema SISDOCT:

Sql.Connection objeto local => con

Sql.Statement objeto local => stmt

Sql.ResultSet objeto local => result

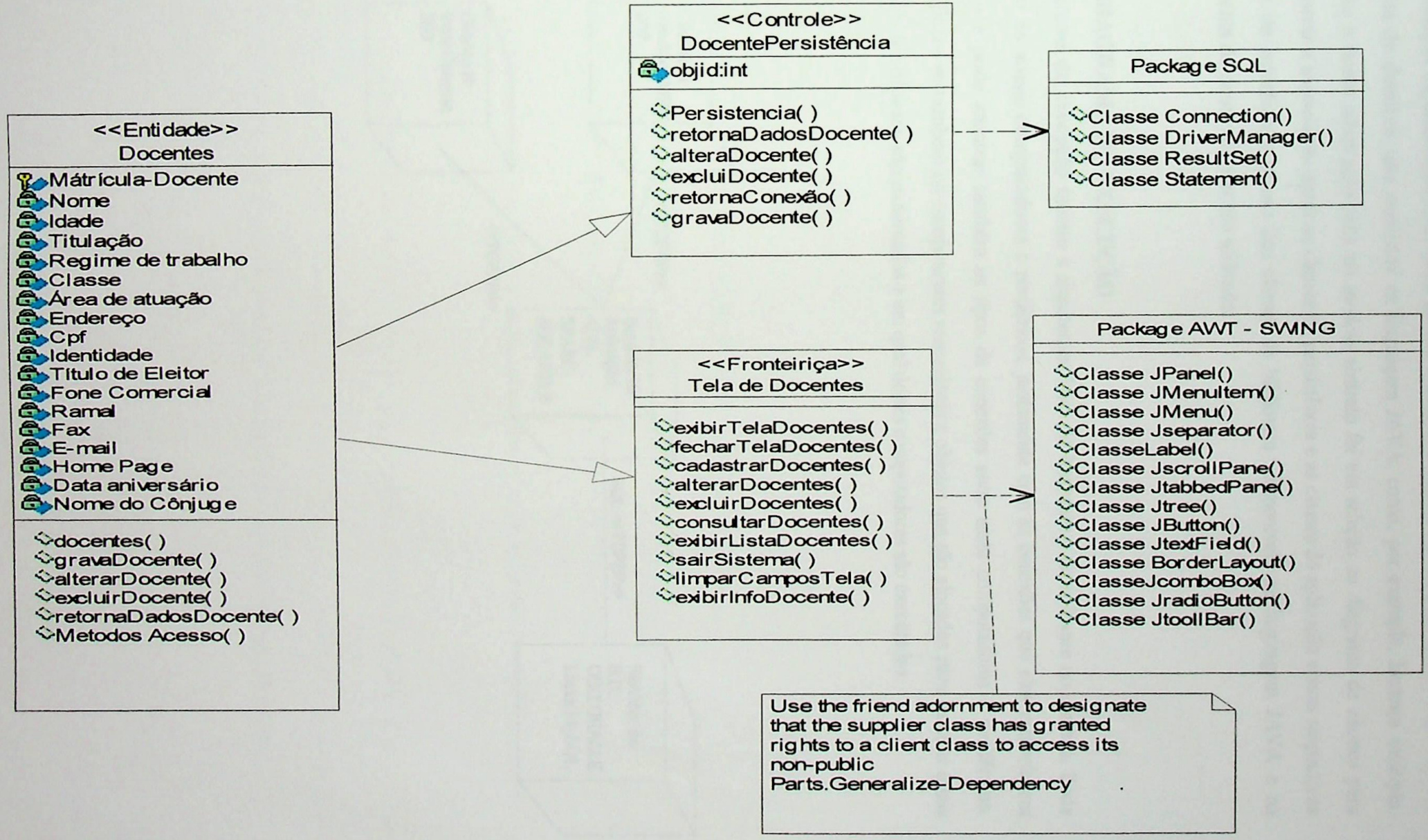
3.33. ADAPTAÇÃO DO MODELO ABSTRATO

Durante essa subfase procuramos ajustar as estruturas conceituais geradas no Modelo Abstrato de acordo com a linguagem alvo escolhida, no caso JAVA.

A partir da análise de reuso anterior, onde foram reusadas diretamente da biblioteca classes contidas nos package SWING, AWT e SQL do JAVA, efetuamos as mudanças correspondentes nos documentos e diagramas contidos no Modelo Abstrato.

A classe DocentePersistencia reusa classes contidas no package SQL, já a classe TelaDocentes reusa classes contidas nos package SWING e AWT. Um novo Diagrama de classes é a seguir apresentado. As demais classes, seguem a mesma idéia das classes Docente Persistência e TelaDocente.

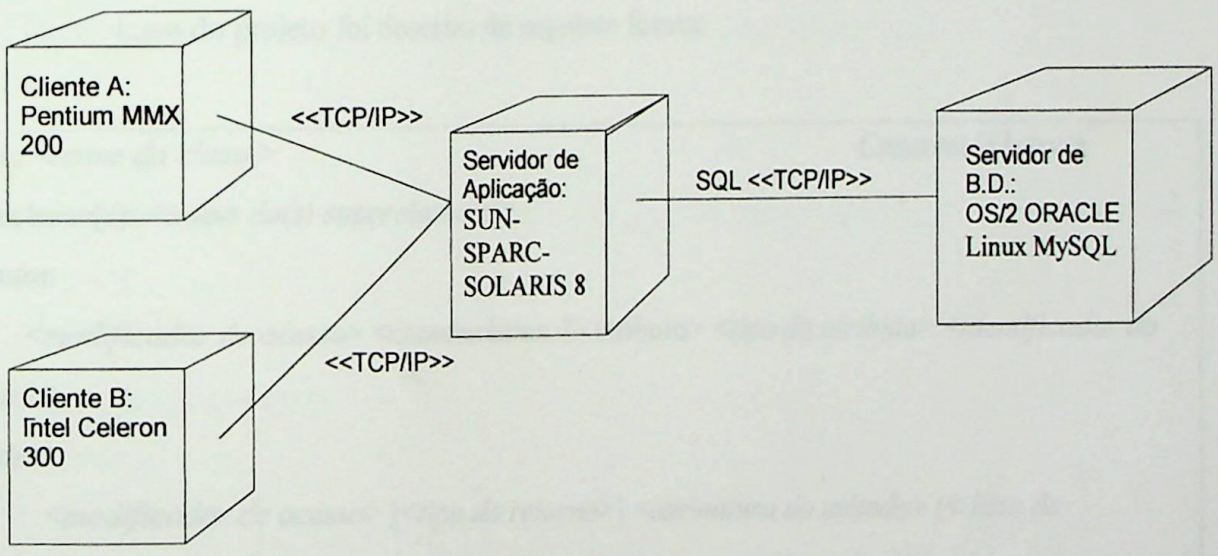
3.34. DIAGRAMA DE CLASSES ADAPTADO – Visão de projeto)



Em função do desacoplamento entre as classes de aplicação, de interface e de persistência, não existe relação alguma entre elas, como mostra o diagrama anterior. O Modelo Abstrato gerado não necessita de detalhes não previstos na linguagem JAVA, como, por exemplo, herança múltipla. Portanto a única adaptação feita no modelo abstrato foi em relação ao diagrama de classes para demonstrar a associação entre as classes de persistência e as classes de aplicação e suas respectivas classes de interface e o uso das classes da biblioteca disponíveis na linguagem JAVA e na ferramenta de desenvolvimento utilizada.

3.35. DIAGRAMA DE EXECUÇÃO

O diagrama de execução mostra a arquitetura física do hardware e do software no sistema. Pode mostrar os atuais computadores e periféricos, juntamente com as conexões que eles estabelecem entre si e pode mostrar também os tipos de conexões entre esses computadores e periféricos. Especificam-se também os componentes executáveis e objetos que são alocados para mostrar quais unidades de software são executados e em qual destes computadores são executados.



3.36. CODIFICAÇÃO E TESTES (Aplicação dos padrões de transição)

Identificados os padrões utilizados no Modelo Abstrato procuramos adequá-los aos exigidos pela Linguagem JAVA. Isso foi feito construindo um Driver da linguagem JAVA que atendesse as nossas necessidades de padronização.

Os padrões que necessitamos são:

- Classe;
- Método;
- Atributo;
- Objeto; e
- Mensagem;

Com o objetivo de facilitar a tradução dos métodos especificados para a linguagem JAVA usamos os padrões a seguir:

Padrão Classe:

Cada classe do projeto foi descrito da seguinte forma:

Classe: *<nome da classe>* ***Concreta/Abstrata***

Superclasse(s): *<nome da(s) superclasse(s)>*

Atributos

<modificador de acesso> <característica do atributo> *<tipo do atributo>* *<identificador do atributo>*;

Métodos

<modificador de acesso> [*<tipo do retorno>*] *<assinatura do método>* (*< lista de argumentos>*);

O cartão CRC é mapeado na linguagem Java, conforme a estrutura apresentada abaixo:

```
[public | friendly | final | abstract] class <nome da classe> extends <nome da superclasse>
{
   // Declaração dos membros de dados da classe
  // Implementação  dos métodos da classe
}
```

Padrão Método:

Na especificação em pseudolinguagem orientada a objetos um método é especificado da seguinte forma geral:

```
// Método em pseudolinguagem orientada a objetos
<modificador de acesso> [<tipo do retorno>] <assinatura do método> (< lista de argumentos>)
inicio
        // Descrição do comportamento do método usando a pseudolinguagem orientada a objetos
fim
```

Cada método da classe Foi convertido de acordo com o esquema a seguir:

```
[public | private | protected | final | static | abstract] [tipo do valor retornado] <nome do método>
(<parâmetros>)
{
   // Declarações de variáveis
    // Comandos
}
```

Padrão Atributo:

Os atributos da classe foram especificados da seguinte forma:

<modificador de acesso> <característica do atributo> *<tipo do atributo>* *<identificador do atributo>*;

Para a linguagem Java usamos a estrutura apresentada abaixo.

[public | private | protected | final | static] [tipo do dado] <nome do atributo>;

Padrão Objeto:

Os objetos foram instanciados conforme o seguinte modelo geral:

<Nome da classe> <identificador do objeto>; // declaração do objeto
<identificador da classe> = nova <nome da classe>(lista de argumento) // instanciação de um objeto da classe

Os objetos são criados em JAVA utilizando o operador *new*.

<Nome da classe> <identificador do objeto>; <identificador da objeto> = new <nome da classe>(lista de argumento)

Padrão Mensagem:

Uma mensagem é enviada a um outro objeto da seguinte maneira a seguir especificada:

// envio de mensagem em pseudolinguagem orientada a objetos <nome do objeto receptor>.< método a ser invocado>(lista de argumentos)

Enviar uma mensagem na linguagem Java pressupõe a existência do nome do objeto, do separador '. '(ponto) do método e de uma lista de argumentos.

<nome do objeto receptor>.<método a ser invocado>(lista de argumentos);

Ao final dessa sub-fase consolidamos os elementos necessários para efetuar a codificação detalhada, que se encontra a seguir:

3.37. CODIFICAÇÃO DETALHADA

```
/**
 * Classe que define a estrutura de um Docente.
 *
 * @author CE224
 * @version 1.0
 * @since 10.11.2000
 * @see Compromisso
 */
```

3.38. PLANO DE TESTES

O teste é elemento de grande importância para o controle de qualidade do software. O tempo a ser gasto no planejamento deve ser visto como investimento. Os testes estão deixando de ser "exterminadores de bugs" para se tornarem parte do processo de maturidade do software. Um teste é um projeto dentro de outro projeto.

A crescente especialização do software como elemento de sistema e o custos associados à falha do mesmo motivam investimentos em planejamentos para teste. Não é raro que uma organização gaste entre 30% a 40% do faturamento, na área de testes [Pressman1997].

O plano de teste se resumiu a um "passeio" completo pelo sistema utilizando as operações previstas. Os use cases foram testados um a um e os resultados plenamente satisfatórios. Os erros ocorridos foram corrigidos e a planilha a seguir mostra o resultado final do plano de testes. Nesta documentação foi usado somente o use case referente a classe Docentes.

Use Cases	Funcionou	Não Funcionou
U1: Usuário cadastra Docentes	✓	
U2: Usuário altera Docente	✓	

U3: Usuário consulta Docente	✓	
U4: Usuário exclui Docente	✓	

Cursos alternativos para os use cases		
U1: Item inválido introduzido. Tela da Interface não permite Inclusão.	✓	
U1: Usuário não confirma inclusão pressionando botão Cancelar da tela de Docentes. Retorna a tela de Docentes apagando o que já foi preenchido.	✓	
U1: O usuário tenta incluir um Docente sem um nome associado. O sistema não habilita o botão salvar.	✓	
U2: Item inválido introduzido. Tela da Interface não permite alteração.	✓	
U2: Usuário desiste das alterações ao utilizar o botão Cancelar da tela de Docentes. Sistema não executa nenhuma alteração dos dados retornando para os valores que existiam anteriormente.	✓	
U2: Usuário não confirma alteração pressionando botão cancelar da tela de mensagem de confirmação de alteração. Sistema cancela transação de alteração de Docente retornando à tela de Docentes e mantendo os valores já alterados.	✓	
U3: Item inválido introduzido. Tela da Interface não permite.	✓	
U3: Usuário pressiona botão finalizar para apagar parâmetros de consulta e visualizar todos os Docentes novamente.	✓	
U4: Item inválido introduzido. Tela da Interface não permite.	✓	
U4: Usuário não confirma exclusão. Sistema cancela transação de exclusão de Docente e retorna a tela de Docentes mantendo os dados do Docente selecionado.	✓	

3.39. ANÁLISE DE RISCO

Risco Não cumprimento do prazo
Estimativa 80%
Impacto Aumento do custo (nota baixa no trabalho)
Alternativa para contornar Aumentar o número de reuniões

Risco Não dominar a Metodologia
Estimativa 30%
Impacto Desobediência ao padrão e falta de qualidade
Alternativa para contornar Buscar orientação para resolver as dúvidas

Risco Não domínio da ferramenta de desenvolvimento por todos os membros do grupo
Estimativa 40%
Impacto Atraso ou não conclusão do projeto
Alternativa para contornar Treinar todos na ferramenta de desenvolvimento

Risco O sistema não se adequar às necessidades do cliente
Estimativa 10%
Impacto Devolução do produto
Alternativa para contornar Marcar mais reuniões com o usuário a fim de captar e validar as suas necessidades

4. CONCLUSÃO

A maturidade de uma organização em Engenharia de Software mede o grau de competência, técnica e gerencial, que essa organização possui para produzir software de boa qualidade, dentro de prazos e custos razoáveis e previsíveis. Em organizações com baixa maturidade em software, os processos geralmente são informais.

A existência de processos definidos é necessária para a maturidade das organizações produtoras de software. Os processos definidos permitem que a organização tenha um *modus operandi* padronizado e reprodutível.

Processos rigorosamente definidos, mas não alinhados com os objetivos da organização são entraves burocráticos, e não fatores de produção.

Para tornar uma organização mais madura e capacitada, é preciso melhorar realmente a qualidade dos seus processos. Processos não melhoram simplesmente por estarem de acordo com um padrão externo.

O critério de verdadeiro êxito dos processos é a medida de quanto eles contribuem para que os produtos sejam entregues aos clientes e usuários com melhor qualidade, por menor custo e em prazo mais curto.

Diversas organizações do mundo propuseram paradigmas para a melhoria dos processos dos setores produtivos; em particular, algumas desenvolveram paradigmas para a melhoria dos processos de software. Estes paradigmas podem assumir diversas formas.

Interessam para a EFEI, especialmente, os paradigmas do tipo modelos de capacitação. Um modelo de capacitação serve para avaliar a maturidade dos processos de uma organização. Um modelo de capacitação particularmente importante para a área de software é o CMM (Capability Maturity Model), do software Engineering Institute. O CMM é patrocinado pelo Departamento de Defesa americano, que o utiliza para a avaliação de seus fornecedores de software. O CMM focaliza os processos, que considera fatores de produção com maior potencial de melhoria a prazo mais curto. www.sei.cmu.edu

Os programas de melhoria de processos devem ser justificáveis através de análises de retorno de investimento. Estas análises procuram medir, para cada unidade monetária investida, quantas unidades monetárias retornam em determinado prazo, através de redução de custos ou do aumento da renda. Existem muitas práticas de Engenharia de Software; os programas de melhoria devem dar prioridade às práticas com melhor retorno de investimento.
Capers Jones, que em [Jones94] havia criticado 0 CMM pela escassez de resultados publicados, relatou os seguintes valores de retorno de investimento, em [Jones98], apresentados abaixo.

Nível do CMM	RI – 1 ano %	RI – 2 anos %	RI – 4 anos %
SEI CMM 2	100	115	250
SEI CMM 3	175	250	500
SEI CMM 4	300	400	950
SEI CMM 5	350	475	1275

Em [Jones98], apresenta-se uma tabela comparativa do retorno de investimento para os principais métodos e técnicas da Engenharia de Software. O retorno de investimento é apresentado como percentagem do investimento inicial, em um prazo médio de quatro anos.

Prática	Retorno do Investimento
Reutilização de artefatos de alta qualidade	4.375
Inspeções de código	1.600
Inspeções de desenho	1.550
Ferramentas de gestão de projetos	1.300
Gestão de configurações (total)	1.200
JAD	1.200
Programação O.O.	1.100
Estruturação de código	1.000
Ferramentas de gestão dos requisitos	950
Métricas de pontos de função	900
Reutilização de código de alta qualidade	700
Teste de regressão	600
Reutilização de desenho	600
Revisões informais de código	550
UML	550
Revisões informais de desenho	500
Teste de unidade	500
Teste de aceitação	500
Reutilização de requisitos	500
Reutilização de planos de projetos	500
Reutilização de casos de teste	500
Desenho O.O.	450

Reutilização de documentação de usuário	375
Gestão de configurações (código)	350
Reutilização de interfaces	350
Prototipagem informal	300
Padrões ISSO e IEEE	300
Métricas de contagem de linhas de códigos	275
Métricas O.O.	170

Analisando agora pelo lado da documentação de Software, a criação de uma linguagem para a comunidade de desenvolvedores em orientação a objetos era uma necessidade antiga. A UML realmente incorporou muitos recursos que dão à linguagem uma extensibilidade muito grande.

A organização da modelagem em visões e a divisão dos diagramas especificando características estáticas e dinâmicas do sistema tornou a UML fácil de ser utilizada e fez com que qualquer tipo de comportamento possa ser visualizado em diagramas.

A modelagem visual orientada a objetos agora possui um padrão, e esse padrão é extremamente simples de ser escrito a mão, sendo robusto para especificar e descrever a grande maioria das funções, relacionamentos e técnicas de desenvolvimento orientado a objetos que hoje são utilizados. Novas técnicas irão surgir e a UML também estará preparada já que tudo estará baseado nas idéias elementares da orientação a objetos.

Sem dúvida alguma a UML facilitará às grandes empresas de desenvolvimento de software uma maior comunicação e aproveitamento dos modelos desenvolvidos pelos seus vários analistas envolvidos no processo de produção de software já que a linguagem que será utilizada por todos será a mesma, acabando assim com qualquer problema de interpretação e mal-entendimento de modelos criados por outros desenvolvedores.

Os modelos criados hoje pela EFEI, poderão ser facilmente analisados por futuras gerações de desenvolvedores acabando com a diversidade de tipos de nomenclaturas de modelos, o grande empecilho do desenvolvimento de softwares orientados a objetos. Adicionalmente os fabricantes de ferramentas CASE agora suportarão a UML em seus softwares e a fase de codificação será cada vez mais substituída pela geração de código automático desempenhada pelas ferramentas CASE.

5. ANEXOS

Os Anexos apresentam o dicionário de dados resultado da Modelagem, duas apostilas: Modelagem Relacional e UML e o Resultado do WorkShop – Sistemas de Informação da EFEI(14 E 15/05/01) Aproximadamente 143 páginas que se encontram no arquivo: Anexos.doc

Referências Bibliográficas:

[Booch1997] Booch, G., Rumbaugh, J., Jacobson, I. Unified Modeling Language Documentation Set 1.0. Santa Clara, CA: Rational Software Corporation. http://www.rational.com. 1997.

[Booch2000] Booch, G., Rumbaugh, J., Jacobson, I. UML, guia do usuário. 4ª Tiragem. Rio de Janeiro: Campus, 2000.

[Brooks75] Brooks, F., The Mythical Man-Month, Addison-Wesley, 1975.

[Chen1990] Chen, Peter. Modelagem de Dados – A Abordagem entidade – relacionamento para Projeto Lógico. 3° Edição. São Paulo McGraw-Hill: Makron Books. 1990.

[Costa1994] Costa, Osvaldo Wilson Dias da. JAD, Join Application Design – Rio de Janeiro. Livr. E Ed. Infobook, 1994.

[Eriksson1998] ERIKSSON, Hans-Erik & PENKER, Magnus. UML Toolkit. Editora Wiley, 1998.

[Gamma2000] Gamma, Erich; Helm, Richard; Johnson, Ralph; Vlissides, John. Padrões de Projeto: Soluções reutilizáveis de software orientado a objetos. Bookman, 2000.

[Jones94] Capers Jones. *Assesment and Control of Software Risks.* Yourdon Press – Prentice-Hall. Upper Saddle River – NJ, 1994.

[Jones98] Capers Jones. The Year 2000 Software Problem. Aaddison-Wesley, Reading – MA, 1998.

[Paula2000] Wilson de Pádua Paula Filho Manual do Engenheiro de Software – Métodos Gerenciais. DCC – UFMG. 2000.

[Pressman1995] Pressman, Roger S. Engenharia de Software, São Paulo, Makron Books, 1995.

[Pressman1997] Pressman, Roger S. Software Engineering, A Practitioners's Approach, forth edition Ed McGraw-Hill, 1997.

[Rumbaugh1994] Rumbaugh, James. Modelagem e Projetos baseados em Objetos, Rio de Janeiro, Campus, 1994.

[Setzer1986] Setzer, Waldemar. Banco de Dados, 1986.

Bibliografia Complementar:

[Booch1994] Booch, G. Object-Oriented Analysis and Design with Applications - Second Edition. Redwood City: CA: Benjamin/Cummings Publishing. 1994.

[Coad1992] Coad, Peter; Yourdon, Edward. Análise Baseada em Objetos, Tradução da Segunda edição americana, Rio de Janeiro, Campus, 1992.

[Coad1993] Coad, Peter; Yourdon, Edward. Projeto Baseada em Objetos, Rio de Janeiro, Campus, 1993.

[Gause1991] Gause, D. Weinberg, Gerald, M. Explorando Requerimentos de Sistemas, Ed. Makron, McGraw-Hill, 1991.

[Jacobson1992] Jacobson, I., Christerson, M., Jonsson, P. Övergaard G.. Object-Oriented Software Engineering - A Use Case Driven Approach. Reading, MA: Addison-Wesley. 1992.

[Jacobson1995] Jacobson, Ivar et. All. Object-Oriented Software Engineering – A use case Driven approach, United States of America: Addison Wesley, 1995.

[Jones1991] Jones, C. Produtividade no Desenvolvimento de Software, Ed. Makron, McGraw-Hill, 1991.

[Martin1994] Martin, James. Princípios de Análise e Projeto Baseados em Objetos, Rio de Janeiro, Campus, 1994.

[North1995] North, David; Mayfield, Mark. Object- Models – Strategies, Patterns and Aplications, YOURDON Press, 1995.

[OMG1991] OMG. Object Management Architecture Guide. Wiley-QED / Object Management Group (OMG). 1991.

[OMG1995] OMG. Common Object Request Broker Architecture and Specification 2.0. Object Management Group. 1995.

[Page1990] Page Jones, Meiller. Gerenciamento de Projetos: Uma Abordagem prática e estratégica no gerenciamento de projetos, São Paulo, MacGraw-Hill, 1990.

[Rumbaugh1991] Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., Lorensen. W. Object-Oriented Modefing and Design. Englewood Cliffs, NJ: Prentice-Hall. 1991.

[Setzer1986] Setzer, Waldemar. Banco de Dados, 1986.

[Sodré1996] Sodré, Eliana. Developers & Object Forum 96: Processo de Desenvolvimento de Sistemas Orientados a Objetos, São Paulo, 1996.

[Weinberg] Weinberg, Gerald M. Software com Qualidade - volumes 1 e 2, Makron Books.

[Winbland1993] Winbland, Ann L. et. All. Software Orientado ao Objeto, São Paulo, Makron Books, 1993.

[Yourdon1990] Yourdon, Edward. Análise Estruturada Moderna. Editora Campus, 1990.

[Yourdon1994] Yourdon, E. Object-Oriented Systems Design - An Integrated Approach. Englewood Cliffs, NJ: PTR Yourdon Press. 1994.

==//==