UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO

(Bacharelado)

AVALIAÇÃO DA QUALIDADE DA FERRAMENTA CASE SYSTEM ARCHITECT BASEADA NA NORMA ISO/IEC 14102

TRABALHO DE CONCLUSÃO DE CURSO SUBMETIDO À UNIVERSIDADE REGIONAL DE BLUMENAU PARA A OBTENÇÃO DOS CRÉDITOS NA DISCIPLINA COM NOME EQUIVALENTE NO CURSO DE CIÊNCIAS DA COMPUTAÇÃO — BACHARELADO

TATIANA MIELE HOFFMANN

BLUMENAU, JUNHO/2001

2001/1-66

AVALIAÇÃO DA QUALIDADE DA FERRAMENTA CASE SYSTEM ARCHITECT BASEADA NA NORMA ISO/IEC 14102

TATIANA MIELE HOFFMANN

ESTE TRABALHO DE CONCLUSÃO DE CURSO, FOI JULGADO ADEQUADO PARA OBTENÇÃO DOS CRÉDITOS NA DISCIPLINA DE TRABALHO DE CONCLUSÃO DE CURSO OBRIGATÓRIA PARA OBTENÇÃO DO TÍTULO DE:

BACHAREL EM CIÊNCIAS DA COMPUTAÇÃO

_____

Prof. Everaldo Artur Grahl — Orientador na FURB

_____

Prof. José Roque Voltolini da Silva — Coordenador do TCC

BANCA EXAMINADORA

_____

Prof. Everaldo Artur Grahl

_____

Prof. Oscar Dalfovo

_____

Prof. Carlos E. Negrão Bizzotto

AGRADECIMENTOS

Aos meus pais, Rubens Heinrich Hoffmann e Karla Kahl Hoffmann, por me apoiarem sempre na vida e nos estudos, pois sem eles, nada disto seria possível.

Gostaria de agradecer a todos os professores do curso de Bacharelado em Ciências da Computação da Universidade Regional de Blumenau, por todo conhecimento adquirido, e em especial ao mestre Everaldo Artur Grahl, pelo apoio e orientação na conduta deste trabalho.

E também gostaria de agradecer ao meu noivo Roberto Silvino da Cunha pela compreensão e carinho durante a realização deste trabalho.

A todos muito obrigado.

SUMÁRIO

LISTA DE FIGURAS

RESUMO

Este trabalho tem como objetivo a avaliação da qualidade da ferramenta CASE *System Architect*. Para esta avaliação foram usadas algumas características previstas na norma ISO/IEC 14102, que trata de avaliação e seleção de ferramentas CASE. Para facilitar a avaliação da ferramenta CASE foi desenvolvido um sistema de auto-avaliação de cursos de Computação. Este desenvolvimento foi realizado a partir de especificações em diagramas da UML (Unified Modeling Language) da ferramenta CASE *System Architect*.

ABSTRACT

This work has as objective the evaluation of the quality of the CASE tool. For this evaluation some characteristics foreseen in the norm ISO/IEC 14102 were used, that is about evaluation and selection of CASE tools. To facilitate the evaluation of the CASE tool a system of evaluation of computer courses was developed. This development was accomplished starting from specifications in diagrams of UML (Unified Modeling Language) of the CASE tool *System Architect*.

1 INTRODUÇÃO

1.1 ORIGEM

As ferramentas CASE *Computer-Aided Software Enginneering* - Engenharia de Software Auxiliada por Computador se propõem a auxiliar no processo de desenvolvimento de sistemas, sob a forma de um suporte computadorizado, aumentando a produtividade de analistas e programadores. Para isto, é imprescindível a definição de uma metodologia, pois quanto mais abrangentes são os padrões, maiores as possibilidades de ganhos em produtividade.

Não existe um consenso ou uma definição formal sobre o que representa a CASE, porém a maioria do pessoal ligado a computação concorda que a mesma, trabalha com todos os processos do ciclo de vida da Engenharia de Software, tais como, especificação de projeto, análise de requisitos e implementação do software, inclusive com a geração automática do programa.

Conforme Fischer (1990), as ferramentas CASE, quando não eliminam, reduzem drasticamente problemas de projeto e implementação, próprios de grandes projetos de software, através da geração automática de código a partir da especificação do projeto e da análise dos requisitos do usuário. Com isso, o desenvolvedor de software, pode concentrar-se mais na arquitetura do sistema, pois a CASE concentra-se mais nessa fase, onde um trabalho bem realizado garante uma geração de código eficaz, enxuta, de qualidade, documentada e de acordo com o que foi projetado.

Um fator primordial que justifica ainda mais o uso de tecnologia CASE é o fato de que os projetos de software vêm crescendo em complexidade, tornando-se propensos a comportamentos imprevisíveis, podendo chegar até ao fracasso total. Esse problema surge devido a enorme quantidade de software necessária para atender aos sistemas de aplicativos desenvolvidos em nossa época atual. Quanto mais crescem os projetos de software, maior a probabilidade de ocorrerem erros, pois deve-se considerar que muitos projetos possuem até alguns milhares de linhas de código, podendo em alguns casos chegar a milhões de linhas.

Para avaliar a qualidade de uma ferramenta CASE podem ser usadas as características previstas na norma ISO/IEC 14102 (ABNT, 2000). Esta norma trata da avaliação e seleção de

ferramentas CASE, cobrindo parcial ou totalmente o ciclo de vida da engenharia de software. Estabelece processos e atividades a serem aplicadas na avaliação de ferramentas. O objetivo desta norma é oferecer ao usuário um caminho que pode ser adaptado de forma a maximizar as chances de sucesso na avaliação e seleção da ferramenta e minimizar os custos e riscos (Weinrich, 1999).

A fim de avaliar a qualidade da ferramenta, foi desenvolvido um sistema de auto-avaliação dos cursos de Computação segundo recomendações do MEC, desde a sua especificação até a geração do código fonte (MEC/SESU, 1997). Este sistema tem como objetivo verificar se os cursos de Computação possuem condições para serem recomendados pelo MEC. A comissão de especialistas de ensino de informática do MEC criou indicadores de qualidade e para cada um deles definiu padrões de qualidade e elaborou um formulário para que as Instituições de Ensino Superior (IES) utilizem de forma estruturada. Desta forma criou uma metodologia de avaliação, dando origem a um “instrumento de avaliação” (MEC/SESU, 1997). Com base neste “instrumento de avaliação”, com os cadastros, formulários, indicadores e padrões de qualidade foi desenvolvido o sistema de auto-avaliação dos cursos de Computação.

Este trabalho focou a orientação a objetos com a *Unified Modeling Language*, que é um padrão de linguagem para especificar, visualizar, documentar e construir artefatos de um sistema. A avaliação da ferramenta CASE foi feita com o *System Architect* da *Popkin Software*. Para a especificação utilizou-se a ferramenta CASE *System Architect* com a modelagem UML.

1.2 OBJETIVOS

O objetivo deste trabalho é avaliar a qualidade da ferramenta CASE *System Architect* utilizando características previstas na norma ISO/IEC 14102, através da implementação de um sistema.

Os objetivos específicos do trabalho são:

a) desenvolvimento de um sistema de auto-avaliação dos cursos de Computação com o uso da ferramenta CASE;

b) analisar os pontos positivos e negativos da ferramenta, a partir da norma ISO/IEC 14102 utilizando a UML.

1.3 ORGANIZAÇÃO

O trabalho está dividido em nove capítulos descritos a seguir.

O primeiro capítulo é a introdução, apresenta uma visão geral do presente trabalho, sua importância e objetivos.

O segundo capítulo descreve as ferramentas CASE, apresenta uma fundamentação sobre ferramentas CASE, a relação entre metodologias e CASE, descreve suas características gerais, vantagens, tipos de ferramentas CASE e encerra o capítulo apresentando algumas formas de avaliar ferramentas CASE.

O terceiro capítulo trata da norma ISO/IEC 14102 - Avaliação e Seleção de ferramentas CASE.

O quarto capítulo trata do *System Architect* e descreve o que a ferramenta suporta.

O quinto capítulo trata da UML e apresenta a UML com todos os seus diagramas.

O sexto capítulo trata da especificação do protótipo, através de diagramas da UML, utilizando a ferramenta CASE *System Architect*.

O sétimo capítulo trata da implementação do protótipo, apresentando algumas telas e relatórios.

O oitavo capítulo trata da avaliação da ferramenta CASE, apresenta através de exemplos, a avaliação da ferramenta CASE em questão.

Por fim mostra-se a conclusão, com os principais resultados do trabalho, suas limitações e possíveis melhoramentos.

2 FERRAMENTA CASE

Este capítulo apresenta uma visão geral sobre ferramentas CASE e aborda a ferramenta CASE *System Architect* que é o foco deste trabalho.

2.1 ORIGEM

Há alguns anos a procura por software vem batendo a demanda por hardware. Antigamente, o custo e a tecnologia do hardware eram bem maiores. Hoje as posições se inverteram. Grande parte dos esforços, recursos humanos, avanços e verbas da área de Informática se concentram no setor de software.

Esta forte procura por software não é recente. Já na década de 60, a explosão da demanda causou o que se chamou "A Crise do Software". Ou seja, ocorreu uma época em que as empresas e departamentos que desenvolviam software gastavam a maior parte de seu tempo e recursos fazendo a manutenção (principalmente correção) do software já desenvolvido. E como aumentava a demanda por novos produtos, os usuários acabavam tendo que esperar muito por produtos que realmente satisfizessem suas necessidades. Esta "crise" levantou o interesse da comunidade de software: era preciso obter maior produtividade e qualidade no desenvolvimento de software. Isto significava aumentar a eficiência e a eficácia das empresas desenvolvedoras de software (**eficiência** significa fazer uma tarefa usando menos recursos possíveis, tais como tempo, recursos materiais, humanos e financeiros; **eficácia** se consegue quando se desempenha uma tarefa e se atinge o objetivo esperado, ou seja, fazer algo e realmente satisfazer a necessidade inicial). A evolução dos ambientes que desenvolvem software não é só uma questão de "atender os pedidos dos usuários" é também "vida ou morte" para as empresas que desejam continuar competindo no mercado. Prova disto, são os certificados de qualidade ISO-9000, distribuídos pela International Standard Organization (uma organização mundial que regula padrões, inclusive de qualidade). Isto tem provocado recentemente uma correria por melhorias de qualidade e produtividade no setor de software (Loh, 1996).

Devido a esta competitividade, as empresas foram obrigadas a desenvolver também software para uso próprio. Inicialmente, era utilizado o termo *"Workbench*", que significa "bancada de trabalho". Ele designava as ferramentas, geralmente automatizadas, que

auxiliavam no trabalho dos desenvolvedores de software. Mais tarde surgiu o termo CASE (Azevedo, 1998)

Conforme Fischer (1990), em estudos realizados na década de 1980, revelaram que 64% dos problemas ocorridos nos softwares tinham sua origem nas fases de especificação de projetos e análise de requisitos, sendo que apenas 30% dessa enorme quantidade de erros era detectada antes que o software passasse para a fase de testes de aprovação. Por outro lado, durante a fase de implementação do projeto, 36% dos erros eram decorrentes de problemas de programação, sendo que 75% desses mesmos erros de codificação eram encontrados antes do início dos testes de aprovação.

2.2 DEFINIÇÃO

O termo *Computer-Aided Software Enginneering* (CASE) significa Engenharia de Software Auxiliada por Computador. Antigamente, as empresas desenvolviam software para outras áreas (como Engenharia, Arquitetura, Medicina, etc) mas não se utilizavam de software para fazer seu trabalho, confirmando o velho ditado "casa de ferreiro, espeto de pau" (Azevedo, 1998)

Segundo Fuggeta (1993) e João (1993), a denominação CASE foi criada no início da década de 1980, quando evidenciou-se a idéia de que ferramentas gráficas, como os Diagramas de Fluxos de Dados (DFD), Modelos de Entidades e Relacionamentos (MER) e Gráficos Estruturados (GE), poderiam ser úteis para as fases de análise e projeto de sistemas.

Uma outra definição é dada por Loh (1996): uma ferramenta CASE é qualquer software que auxilia as pessoas (desenvolvedores e analistas) que trabalham numa empresa. A presença de ferramentas CASE é vital hoje em dia para o bom funcionamento de uma empresa desenvolvedora de software. Elas existem auxiliando todo o ciclo de desenvolvimento (análise, projeto, implementação e teste) e são também de suma importância para a manutenção do software. Há também ferramentas CASE para apoiar a gerência dos projetos de desenvolvimento.

Uma definição simples e ao mesmo tempo completa é dada por (McClure, 1989), CASE é a automação do desenvolvimento de software. Nessa concepção, CASE oferece aos desenvolvedores de software uma abordagem diferente para o ciclo de vida de software,

sendo essa, baseada na automação. A idéia básica é a existência de um conjunto integrado de ferramentas que possibilitam a economia de trabalho, unindo e automatizando todas as fases do ciclo de vida do software.

2.3 METODOLOGIAS DE DESENVOLVIMENTO DE SISTEMAS

Ferramentas CASE implementam e dão suporte a pelo menos uma metodologia de desenvolvimento de sistemas. A maioria das ferramentas atuais implementam a metodologia de Análise Estruturada e a metodologia de Análise Orientada a Objetos. Devido a essa realidade, nota-se que tentar adotar a tecnologia de ferramentas CASE sem que a organização possua uma sólida fundamentação em metodologias de desenvolvimento de software, é bastante improvável que se obtenha sucesso e os benefícios que a CASE pode oferecer.

A fim de melhor entender a união entre as metodologias desenvolvidas durante a história da computação e conseqüentemente sua adoção por parte das ferramentas CASE, faz-se necessário conhecer a própria evolução dessas metodologias. Uma evolução cronológica aproximada das metodologias de desenvolvimento de sistemas é apresentada por Nascimento (1993), no quadro 2.1. Através da análise da tabela, verifica-se que a bastante tempo existe a preocupação em se desenvolver e utilizar metodologias, visto que por volta do ano de 1965, desenvolveu-se a metodologia de Análise Estruturada. Durante um longo período, essa metodologia dominou o cenário de desenvolvimento de software, até que por volta do ano de 1990, evidenciou-se com maior ênfase, a utilização da metodologia de Análise Orientada a Objetos.

QUADRO 2.1: Evolução das metodologias de desenvolvimento de sistemas

Ano (época)	**Metodologias (técnicas) de desenvolvimento de sistemas**
1965	Metodologia estruturada
1970	Técnicas de modelagem de dados
1975	Projeto de bancos de dados Bancos de dados Linguagens de quarta geração
1980	Especificação do projeto Ferramentas de software Modelagem de dados Linguagens de quarta geração Prototipação
1985	Interface com o usuário Ferramentas de prototipação Automação das metodologias (ferramentas CASE)
1990	Ferramentas de geração de código Metodologias orientadas a objetos

FONTE : Nascimento (1993)

Fator importante é escolha de uma metodologia de desenvolvimento de sistemas, pois ela poderá se revelar inapropriada para automação através do uso de ferramentas CASE. Caso a metodologia escolhida seja manual, o descontentamento com o uso da metodologia poderá migrar também para o uso da CASE e a organização não aproveitará todos os benefícios que resultam da escolha de uma metodologia adequada juntamente com ferramentas CASE que possibilitem implementar tal metodologia.

2.4 CARACTERÍSTICAS GERAIS

Conforme Fischer (1990), as ferramentas CASE devem realizar as seguintes tarefas:

a) fracionamento da complexidade: uma das metas principais da tecnologia CASE é decompor os requisitos e os projetos em componentes manejáveis. Sua função é simplificar, explicar e reduzir;

b) adequação a um público diversificado: para as fases de requisitos e de projeto do ciclo de software, as ferramentas CASE servem a diversos mestres. Por um lado, sua saída deve ser inteligível para os usuários finais e as organizações contratantes, que pagam pelo desenvolvimento do software. Por outro lado, devem oferecer uma ajuda valiosa aos próprios fomentadores; caso contrário, é perda de tempo utilizá-las;

c) mais baratas que a construção em si: a utilização de uma ferramenta CASE deve custar menos e ser mais eficaz a longo prazo que a montagem de um sistema de software pelos métodos tradicionais. As ferramentas CASE devem reduzir substancialmente o empenho despendido em implementação e manutenção, oferecendo especificações e projetos de qualidade superior;

d) quantitativas e verificáveis: as especificações e projetos gerados pelas ferramentas CASE devem articular meticulosa e concisamente as características e os componentes de software a ser construído. Cada exigência da implementação tem que ser verificável, e poder ser encontrada no documento dos requisitos. Os critérios de desempenho, as limitações e as condições de erro devem estar especificadas no projeto;

e) de fácil manutenção: as especificações e projetos produzidos ou montados por uma ferramenta CASE devem ser adaptáveis às modificações nas metas dos requisitos e dos projetos. Quando um documento do projeto perde a sincronização com o código subordinado, torna-se inútil, e pode até causar perda de tempo aos fomentadores em futuros aperfeiçoamentos do software;

f) orientação gráfica: as boas ferramentas CASE apresentam informações visuais de especificação e projeto. São para a engenharia de software o que os programas CAD são para o projeto esquemático e o layout. Tanto para os usuários finais como para os fomentadores, é muito mais fácil compreender uma ilustração gráfica do que ler inúmeras páginas de texto descritivo.

2.5 CATEGORIZAÇÃO

Segundo Nascimento (1993), há poucos anos, poucos analistas de sistemas tinham conhecimento do termo CASE. Agora é o centro das atenções, e o que há de mais recente e atual para reportar os problemas próprios de desenvolvimento de sistemas. O mercado de ferramentas CASE tem um dos mais altos graus de crescimento das empresas que atuam no setor de desenvolvimento de softwares. Como o mercado nasce, a definição de CASE continua a mudar e não há um padrão definido para a sua categorização. Abaixo, Nascimento (1993) identificou alguns termos que melhor identificam esta categorização:

a) *front end* ou *upper CASE*: são aquelas que apóiam as etapas iniciais de criação dos sistemas, são elas as fases de planejamento, análise e projeto do programa ou aplicação (ou seja, a parte lógica);

b) *back end ou lower CASE*: são aquelas que dão apoio a parte física, isto é, a codificação, testes e manutenção da aplicação, normalmente encontram-se produtos de ambos os lados, mas o difícil é se obter uma compatibilidade entre ambas, isto é, conseguir com o auxílio de uma única ferramenta (ou mais de uma, de forma integrada);

c) *I-CASE* ou *integrated CASE*: classifica os produtos que cobrem todo o ciclo de vida do software, desde os requisitos do sistema até o controle final da qualidade;

Outra classificação nos é dada por Nascimento (1993), segundo a fase do ciclo de vida que apoia:

a) *planning workstation*: uma estação de trabalho que automatiza planejamento de sistemas da empresa. Consiste de editores de matrizes e dicionário de dados que apóiam técnicas de planejamento estratégico de informações;

b) *analysis workstation*: apóia o trabalho do analista de sistemas, auxiliando-o no trabalho de especificação, rigorosa e detalhadamente, sistemas de aplicação segundo as técnicas estruturadas de análise de sistemas. Este é o tipo de estação de trabalho com maior desenvolvimento e inovação. O próprio termo CASE está, em grande parte, ligado a esse tipo de software;

c) *design workstation*: voltada para o projetista de sistemas, possui editores gráficos capazes de auxiliar no projeto da arquitetura de código dos sistemas, de acordo com métodos que variam entre Yourdon-Constantine, Warnier, Jackson e James Martin;

d) ferramentas de programação: são ferramentas de programação CASE para suportar a implementação de programas. Muitas destas ferramentas são familiares e amplamente usadas pelos programadores. A diferença é que esses pacotes de ferramentas são feitos para se integrarem com outros.

2.6 VANTAGENS

Para a grande maioria das organizações de desenvolvimento de software, as vantagens qualitativas obtidas através do uso de ferramentas CASE tem um peso maior que as vantagens quantitativas. Segundo Fischer (1990), o tempo gasto no desenvolvimento será quase sempre menor com o auxílio das ferramentas CASE, mas talvez seu maior benefício tenha a forma de garantia, a consciência tranqüila de que a tarefa está sendo realizada devidamente como foi programada e seguindo as especificações do usuário.

Segundo Fischer (1990) e Chikofsky (1992), entre as diversas vantagens que se obtém com o uso de ferramentas CASE, destacam-se:

a) especificações completas dos requisitos: especificar completamente os requisitos do sistema é o objetivo das ferramentas CASE de análise e de especificação dos requisitos. A maioria das metodologias de especificação impõe o envolvimento do usuário final com o objetivo de desenvolver um modelo. Embora ainda exista o risco de se criar uma grande confusão, apesar do maior empenho de todos os envolvidos, essa probabilidade é bem menor, se houver uma especificação completa, detalhada e minunciosa dos resultados;

b) especificações minunciosas do projeto: documentar as especificações do projeto de forma completa e precisa é necessário para o entendimento, desenvolvimento e manutenção do software. Além disso, a documentação deve recomendar a separação de temas com módulos distintos, cada um com entradas e saídas bem definidas, onde cada módulo é tratado como uma caixa preta, cuja estrutura interna

de implementação não é conhecida do público. Apenas os parâmetros de entrada e saída são visíveis aos demais módulos;

c) especificações atuais do projeto: as ferramentas CASE para projetos ajudam a manter uma sincronia com a implementação do código. Muitas delas vinculam de fato o código à especificação, de modo que, se a especificação se modificar, o código a ela subordinado também se modifique. Para as ferramentas CASE atuais, que na sua maioria geram código automaticamente, não há necessidade de se manter a sincronização, pois o desenvolvedor não toca no código, apenas edita o projeto do sistema e a implementação é gerada ou regerada automaticamente. As ferramentas CASE oferecem dispositivos para manterem as especificações do projeto atualizada em fase de manutenção contínua do software, redefinição de projeto, aprimoramento e evoluções de implementação;

d) redução do tempo de desenvolvimento: a especificação completa da arquitetura do software visa eliminar a perda de tempo em implementações equivocadas, sendo que esse ganho de produtividade se traduz em tempo de implementação reduzido. Por causa da orientação gráfica altamente interativa da maioria das ferramentas CASE, utilizar uma delas leva muitos técnicos de software a acreditar que estão escrevendo o código, quando na realidade estão fazendo os projetos arquitetônicos do software. O tempo na elaboração do projeto é mais do que compensado nas fases de implementação, testes e lançamento;

e) código altamente flexível e de fácil manutenção: nenhum projeto de software bem sucedido chega ao fim. Os usuários finais vão exigir ou sugerir aperfeiçoamentos funcionais ou identificarão erros na operação do software, tornando necessária alguma forma de desenvolvimento contínuo ou de serviço de manutenção. Embora os pequenos aperfeiçoamentos ou reparos de erros possam não garantir uma atualização da especificação do projeto, a combinação de diversas assistências na manutenção das especificações do projeto, se o software desenvolvido a partir dessas especificações estiver escrito à mão a partir da geração automática da implementação pela própria ferramenta. Felizmente, ferramentas que geram código automaticamente a partir da especificação do projeto de software, não se atrapalham com esse problema.

Conforme Weinrich (1999), quando se descreve as vantagens do uso da tecnologia CASE, corre-se sempre o risco de que alguma facilidade, vantagem ou característica importante dessas ferramentas fique à margem das observações e descrições. Uma importante justificativa para o uso de ferramentas CASE, ressaltando a vantagem de seu uso, é apresentada por Fournier (1994), que recomenda firmemente o uso de diversas ferramentas automatizadas ao longo das fases do ciclo de vida de desenvolvimento e manutenção. Na realidade, essa é uma tendência predominante nas áreas onde o software pode ser gerado automaticamente, como no caso do projeto de interface com o usuário e no de acesso à banco de dados.

No entanto, nem tudo são aspectos positivos, e estas vantagens podem mesmo ser contrariadas por alguns problemas. No passado, verificou-se freqüentemente que os resultados obtidos não estavam de acordo com as elevadas expectativas criadas, o que provocou o fracasso da introdução das ferramentas CASE nas organizações. Um dos fatores mais referidos como responsável por esta situação é o elevado tempo de aprendizagem, por vezes requerido para tirar o melhor partido destas ferramentas, e que não é compatível com as exigências das organizações apresentarem resultados o mais rapidamente possível. Uma área que ainda hoje constitui um dos principais mitos do software é a geração automática de código: desde há muito tempo tem sido um dos principais objetivos que se procura atingir com as ferramentas CASE, mas onde ainda não foram obtidos resultados completamente satisfatórios. (Silva; Videira, 2001).

2.7 AVALIAÇÃO

Alguns autores têm abordado a questão de avaliação de ferramentas CASE. No livro de Fischer (1990), são apresentadas algumas questões para avaliação de CASE:

A ferramenta tem um escopo bem definido ? Após definidos os requisitos da organização, deve-se proceder à escolha de ferramentas que os satisfaçam, não se deixando levar pelo apelo dos fornecedores e sim, pelo atendimento integral dos requisitos definidos;

A ferramenta poderá ser personalizada se necessário ? Algumas organizações desenvolveram extensões das metodologias padrões existentes ou desenvolveram

metodologias originais e necessitam acrescentar seus próprios símbolos e regras para obter da ferramenta a automação seguindo seus princípios e práticas de desenvolvimento;

A integração é suficiente para auxiliar as tarefas de projeto e desenvolvimento de sistemas ? A ferramenta precisa oferecer uma vantagem maior do que as resultantes dos métodos atualmente empregados;

A ferramenta gera software automaticamente a partir da especificação do projeto? A geração automática do software é uma característica amplamente procurada nas ferramentas CASE atuais, com a garantia de que o software gerado seja isento de erros e de fácil manutenção, também realizada pela ferramenta.

Conforme Nascimento (1993), são citados alguns aspectos importantes sobre a avaliação de ferramentas CASE que devem ser verificados, tais como:

Capacidade de gerar protótipos: Uma importante característica que agiliza o desenvolvimento de software, bem como sua interação com o usuário final, é a capacidade de se desenvolver protótipos que tem por finalidade verificar se os requisitos do usuário estão sendo ou não atendidos;

Capacidade de documentar o projeto: A documentação do projeto e da implementação do software constituem fator básico na avaliação de ferramentas CASE devido à sua importância para o registro das atividades desenvolvidas e facilidade do trabalho em futuras manutenções do projeto e do software;

Permitir distribuição de atividades: A ferramenta deve permitir que diversos profissionais atuem em diferentes áreas tanto de projeto quanto de implementação e facilitar sua completa integração e controlar a execução dessas mesmas atividades;

Facilidade de manter o projeto: Não é suficiente facilitar a manutenção do software, mas principalmente manutenção no projeto que constitui a base para a implementação do software.

Neste trabalho o enfoque adotado para a avaliação de ferramentas CASE foi a utilização parcial da norma ISO/IEC 14102. Maiores detalhes sobre esta norma podem ser visto no capítulo três.

2.8 SYSTEM ARCHITECT

O *System Architect* é uma ferramenta desenvolvida pela empresa *Popkin Software*. Surgiu inicialmente no mercado como ferramenta de modelagem de software, recorrendo a técnicas estruturadas, quer de modelagem de processos quer de dados. Ao longo do tempo, o *System Architect* foi evoluindo e incorporando outras notações (modelagem de negócio e modelagem orientada a objetos). Suas últimas versões passaram a incluir também a linguagem UML, sem nunca deixar de suportar as anteriores notações. É natural que a posição de destaque que tinha assumido no mercado das ferramentas de modelagem que utilizam abordagens estruturadas, lhe permita desempenhar um papel importante nas ferramentas de modelagem em UML (Silva; Videira, 2001).

A nível das técnicas de modelagem, as principais funcionalidades disponibilizadas pela ferramenta podem ser agrupadas do seguinte modo:

- modelagem de processos de negócio;
- modelagem de software segundo abordagens estruturadas;
- modelagem de dados, com possibilidade de geração de esquemas de bases de dados e *reverse engineering*;
- modelagem de software segundo abordagens orientadas a objetos, particularmente a UML;
- repositório configurável e extensível;
- suporte ao trabalho em equipes.

O System Architect apóia a maioria das áreas de modelagem:

- do processo do negócio;
- de dados relacionais;
- de análise e projetos estruturados;

- modelagem OO e de componentes com uso da UML (também suporta outras metodologias OO (mais antigas): OMT, Booch, Shlaer/Mellor, Coad/Yourdon);

Além disso o *System Architect* permite modelar e documentar todo o processo de análise e projeto desde a captura dos requisitos, através da :

- análise em nível de negócio – uso de diagramas use cases da UML;

- análise da dinâmica do sistema – através dos diagramas de seqüência, colaboração e estados da UML.

O *System Architect* tem ainda como característica importante o fato de ser uma ferramenta *workgroup*, ou seja, é possível compartilhar um mesmo projeto entre diversos analistas de desenvolvimento. Num único repositório são colocadas todas as informações do projeto.

Como a maioria das ferramentas de modelagem de software, que suportam múltiplas notações e técnicas de modelagem, o *System Architect* não privilegia a utilização de nenhuma metodologia. Esta flexibilidade permite que, num mesmo projeto, seja possível utilizar técnicas de diferentes abordagens, em função da parametrização efetuada quando se cria um novo projeto (Silva; Videira, 2001).

Na fase de projeto o *System Architect* possibilita integrações com banco de dados através de engenharia direta ou reversa dos dados.

A partir do modelo entidade - relacionamento, o *System Architect* gera scripts (SQL) para a maioria dos bancos de dados relacionais do mercado, de modo que o SGBD cria as tabelas a partir deste script ou via ODBC, onde o *System Architect* cria as tabelas físicas diretamente no banco de dados.

A partir do modelo entidade - relacionamento, é possível gerar telas com operações de inserção , deleção , alteração, e consulta , para Visual Basic, Delphi e Power Builder (neste trabalho será utilizada a linguagem Delphi).

O objetivo dessas ferramentas é que possam agregar produtividade na fase inicial do projeto preparando-o, de forma adequada, para a fase de programação da aplicação com o *front-end* de mercado da preferência do desenvolvedor.

2.8.1 SYSTEM ARCHITECT E A UML

Segundo Silva e Videira (2001), o *System Architect* suporta a linguagem UML, com destaque para as seguintes características:

- captura de requisitos e modelagem de cenários através de casos de uso;
- modelagem da dinâmica do sistema com os diagramas de seqüência e de colaboração;
- construção de diagramas de classes de forma iterativa, e modelagem do comportamento das classes através dos diagramas de estado;
- geração de código, para diversas linguagens, a partir dos diagramas de classes e recorrendo a um gerador de código configurável;
- modelagem do comportamento dos componentes;
- modelagem da instalação do software.

A UML será mais detalhada no quinto capítulo. Para este trabalho foi adotada a especificação da UML com ferramentas CASE.

2.9 NORMA ISO/IEC 14102- AVALIAÇÃO E SELEÇÃO DE FERRAMENTAS CASE

Este trabalho trata da norma ISO/IEC 14102 em função de ser a base para a avaliação da ferramentas CASE.

Com o propósito de servir como referencial para aqueles que desejam avaliar e selecionar ferramentas CASE, a Associação Brasileira de Normas Técnicas (ABNT) criou no Brasil em setembro de 1996, a Comissão de Estudos CE-21:101.05 de Avaliação e Seleção de

Ferramentas CASE – referente a norma ISO/IEC 14102, subordinada à Comissão Técnica de Engenharia de Software e Portabilidade.

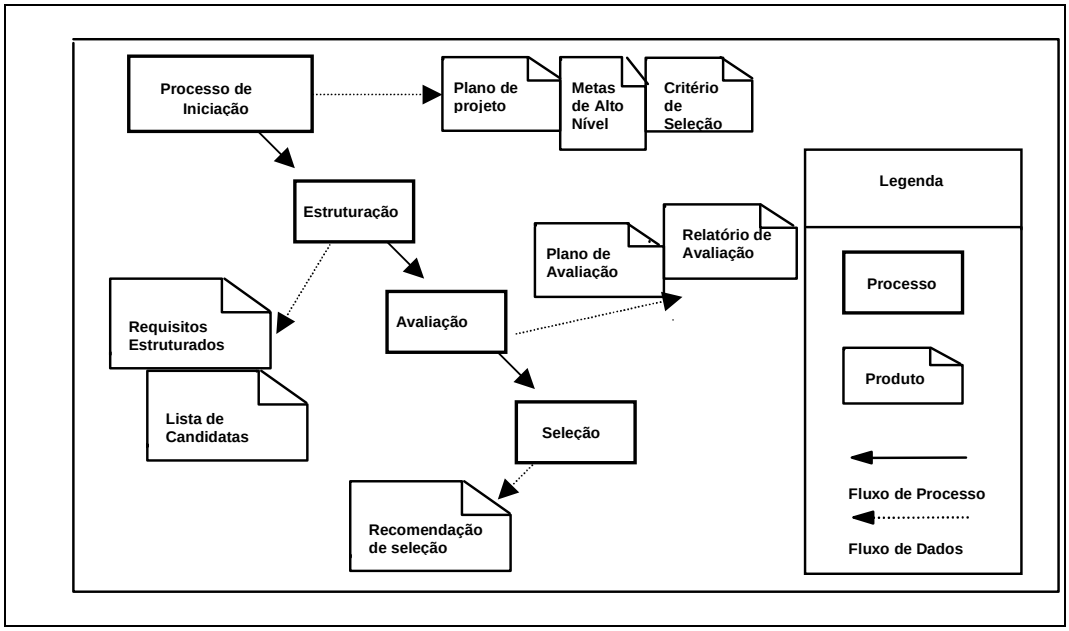
A norma, trata de avaliação e seleção de ferramentas CASE, cobrindo parcial ou completamente o ciclo de vida da engenharia de software. Estabelece processos e atividades a serem aplicadas na avaliação de ferramentas e na seleção da ferramenta mais apropriada dentre várias candidatas. Estes processos são genéricos e as organizações devem adaptá-los de acordo com suas necessidades (ABNT, 2000).

Além de uma avaliação técnica responder o quanto a ferramenta CASE atende aos pré-requisitos de seus usuários, responde também a questão do quanto a ferramenta atende à funcionalidades requeridas. O objetivo do processo de avaliação técnica é apresentar resultados quantitativos nos quais a seleção final possa ser baseada. A mensuração atribui números para as características da ferramenta sendo a principal atividade da avaliação, obter estes valores para a avaliação. Os resultados da seleção final devem buscar objetividade, repetibilidade e imparcialidade. Estes objetivos e a credibilidade dos resultados dependerão em parte dos recursos alocados ao processo global de avaliação e seleção.

Para serem amplamente aceitos, esses processos de avaliação e seleção devem ter utilidade para usuários e fornecedores de ferramentas *CASE* e para a comunidade em geral. As informações apresentadas na norma deverão levar a seleções mais eficientes em termos de custos e a uma maior uniformidade na descrição de funções e características de ferramentas *CASE*.

Segundo a ABNT (2000), para a avaliação e seleção de ferramentas CASE, faz-se necessário a implementação de quatro processos que são o processo de iniciação, estruturação, avaliação e seleção. Dentre esses, um processo chave é a estruturação, composto de um conjunto de requisitos sob os quais as ferramentas CASE candidatas serão avaliadas e servirão como base para decisões de seleção. A figura 2.1, apresenta uma visão geral da avaliação e seleção de ferramentas CASE a partir da norma ISO/IEC 14102.

FIGURA 2.1: Apresenta a visão geral



FONTE: ABNT (2000)

2.10 CRITÉRIOS PARA AVALIAÇÃO DA CASE

A avaliação da ferramenta CASE neste trabalho será feita através de algumas características previstas na norma ISO/IEC 14102. A seleção destas características foi realizada em conjunto com o orientador do trabalho. Foram selecionadas da norma algumas características com suas sub-características que estão detalhadas neste item. A seguir serão citadas as características e sub-características utilizadas para a avaliação:

a) Características relacionadas ao processo de ciclo de vida (modelagem, construção, manutenção, documentação).

Conjunto de atributos que evidenciam a existência de um conjunto de funções e suas propriedades especificadas para suportar o uso de ferramentas CASE, relacionado ao processo e atividades do ciclo de vida da engenharia de software. Para estes processos de ciclo de vida referenciados, as definições na NBR ISO/IEC 12207 se aplicam.

- Desenvolvimento de diagrama: atributos relacionados à habilidade de suportar a entrada e edição de tipos de diagramas de interesse do usuário e a tradução entre tipos de diagramas e entre diagramas e texto.

- Análise de diagrama: atributos relacionados à habilidade de suportar a análise de figuras gráficas, entradas para a ferramenta *CASE* e requisitos de extração e armazenamento e/ou informação de projeto.

- Prototipação: atributos relacionados à habilidade para gerar um modelo de protótipo de todo ou parte de um sistema, baseado em requisitos fornecidos pelo usuário e/ou informações de projeto.

- Geração de Código: atributos relacionados à habilidade de gerar códigos em uma ou mais linguagens específicas, baseado nos dados de projeto disponíveis para a ferramenta *CASE*.

- Geração de Esquemas de Banco de Dados: atributos relacionados à habilidade de gerar esquemas de banco de dados, baseado em informações fornecidas pelo usuário.

- Geração de Telas: atributos relacionados à habilidade de gerar telas de apresentação baseado em informações fornecidas pelo usuário.

- Geração de Relatórios: atributos relacionados à habilidade de automatizar o desenvolvimento de relatórios a serem produzidos pelo sistema em desenvolvimento (em oposição à ferramenta CASE).

- Engenharia Reversa de Dados: atributos relacionados à habilidade de extrair informação do código fonte, o qual define ou descreve os elementos de dados e estruturas do software.

- Suporte para Hipertexto: atributos relacionados à habilidade de suportar formatos e funções de hipertexto.

- Extração Automática de Dados e Geração de Documentação: atributos relacionados à habilidade de aceitar, armazenar e recuperar especificações de conteúdo, formato e layout de dados textuais e gráficos e sua habilidade de, então, extrair e produzir os dados em conformidade com uma especificação.

b) Características relacionadas ao uso da ferramenta CASE (Ambiente, Integrabilidade, Aplicação).

As seguintes características relacionam a ferramenta ao seu ambiente e aos projetos que irão suportar:

- Características de Hardware Requerido pela Ferramenta: atributo relacionado a quaisquer requisitos de hardware para a sua utilização.

- Ambiente de Software Requerido pela Ferramenta: atributo relacionado a quaisquer itens de software que sejam requeridos para a sua utilização.

- Repositório de Software (Base de Informação): atributo relacionado à habilidade para alojar e administrar todas as informações relevantes do processo de engenharia de software. Estes incluem sua habilidade de criar informações desenvolvidas em uma atividade do ciclo de vida para uso durante outras atividades, assim como sua habilidade para prover acesso às informações a outros elementos do ambiente.

- Compatibilidade com Elementos do Ambiente: atributo relacionado à habilidade de interoperar e/ou trocar dados com ambientes de hardware/software.

- Integração de Dados: atributo relacionado à sua habilidade de utilizar, processar e distribuir informação compartilhada por outras ferramentas ou parte de um repositório.

- Ambiente de Hardware e Software dos Produtos da Ferramenta: atributos relacionados ao conjunto de itens de hardware e software com os quais os produtos da ferramenta podem ser usados.

- Tamanho de Aplicação Suportado: atributos que vão resultar em limitações de tamanho da aplicação e, conseqüentemente, em limite de aplicabilidade da ferramenta.

- Linguagens Suportadas: atributos relacionados à habilidade de suportar linguagens específicas.

- Bancos de Dados Suportados: atributos relacionados à habilidade de suportar bancos de dados específicos.

- Suporte Metodológico: atributos relacionados ao conjunto de métodos ou metodologias que podem ser suportados.

c) Características gerais de qualidade (Funcionalidade, Confiabilidade, Usabilidade, Eficiência, Portabilidade).

As seguintes características descrevem a qualidade da ferramenta em termos da NBR 13596 (ISO/IEC 9126):

- Segurança: atributos relacionados à habilidade de evitar uso não autorizado ou inadequado da ferramenta.

- Conformidade técnica: atributos relacionados a aderência ou conformidade a quaisquer padrões específicos.

- Integridade de Dados: atributos relacionados com a capacidade de armazenar e recuperar informações corretamente com um elevado grau de confiabilidade.

- Tolerância à falhas: atributos relacionados à habilidade de manter um nível de desempenho especificado (ex: capacidade reduzida) em casos de falhas diversas (ex: falhas de hardware, software ou de rede).

- Recuperabilidade: atributos relacionados à habilidade de restabelecer seu nível de desempenho e recuperar os dados que sofreram danos no momento da falha e o tempo e esforço para tal.

- Inteligibilidade: atributos relacionados à habilidade de se integrar nas atividades dos usuários, levando em consideração o nível de experiência e especialização, bem como os conceitos, informações, representações e procedimentos que fazem parte do domínio e da cultura profissional e pessoal do usuário.

- Operacionalidade: atributos relacionados com funcionalidades que permitem ao usuário conhecer o estado de operação da ferramenta, estabelecer o relacionamento

de causa e efeito entre suas ações e o estado da ferramenta, e direcionar as ações do usuário.

- Manipulação de Erros: atributos relacionados à habilidade para auxiliar e guiar seus usuários em identificar e corrigir erros, e em manter a integridade da ferramenta (evitando dados incorretos e mudanças do processo).

- Facilidade de Aprendizagem: atributos relacionados com o tempo e esforço necessários para um usuário entender as operações básicas da ferramenta e se tornar produtivo.

- Qualidade da Documentação: atributos relacionados com a qualidade geral da documentação fornecida com a ferramenta.

- Facilidade de Instalação: atributos relacionados à facilidade do usuário realizar a instalação inicial da ferramenta e de suas subseqüentes atualizações.

- Tempo de resposta: atributos relacionados ao desempenho da ferramenta ao realizar tarefas definidas.

- Requisitos de Armazenamento de Dados: atributos relacionados ao espaço necessário para armazenar a própria ferramenta e qualquer base de dados gerada pela mesma.

- Capacidade de Memória Aceitável: atributos relacionados à quantidade de memória endereçável necessária para carregar e operar a ferramenta.

- Portabilidade para Diferentes Plataformas de Hardware: atributos relacionados com a capacidade para executar sobre várias versões de hardware da mesma plataforma ou diferentes plataformas de hardware.

- Compatibilidade entre Diferentes Sistemas Operacionais: atributos relacionados com a capacidade para executar sobre várias versões do mesmo sistema operacional ou sobre sistemas operacionais diferentes, e a facilidade da ferramenta de ser modificada para executar sobre atualizações de um sistema operacional.

d) Características gerais não relacionadas a qualidade (Suporte, Certificação)

As características a seguir são genéricas por natureza e endereçam a própria ferramenta e o desenvolvedor/fornecedor:

- Perfil do Fornecedor: atributos relacionados a indicações gerais de características do fornecedor.

- Perfil do Produto: atributos relacionadas a informações gerais sobre o uso do produto.

- Disponibilidade de Treinamento: atributos relacionados à disponibilidade de materiais e cursos de treinamento, tanto nas instalações do fabricante como do comprador.

Não foram utilizadas para esta avaliação as sub-características: processo de gerenciamento, processo de gerenciamento da configuração, processo de garantia da qualidade, processo de verificação, processo de validação, integrabilidade da ferramenta CASE e processo de aquisição.

3 UML

Neste capítulo são apresentados conceitos e definição sobre a UML e como a ferramenta CASE *System Architect* suporta esta abordagem.

3.1 INTRODUÇÃO

Conforme citado por Barros (1998), 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. Cada simbologia existente possui seus próprios conceitos, gráficos e terminologias, resultando numa grande confusão, especialmente para aqueles que querem utilizar a orientação a objetos não só sabendo para que lado aponta a seta de um relacionamento, mas sabendo criar modelos de qualidade para ajudá-los a construir e manter sistemas cada vez mais eficazes.

Quando a UML foi lançada, muitos desenvolvedores da área da orientação a objetos ficaram entusiasmados já que essa padronização proposta pela UML era o tipo de força que eles sempre esperaram. A UML é muito mais que a padronização de uma notação. É também o desenvolvimento de novos conceitos não normalmente usados. 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 (Barros, 1998).

3.2 DEFINIÇÃO

Conforme Furlan (1998) a UML é uma linguagem padrão para especificar, visualizar, documentar e construir artefatos de um sistema e pode ser utilizada com todos os processos ao longo do ciclo de desenvolvimento e através de diferentes tecnologias de implementação. Buscou-se unificar as perspectivas entre os diversos tipos de sistemas e fases de desenvolvimento de forma que permitisse levar adiante determinados projetos que antes não eram possíveis pelos métodos existentes.

Uma outra definição é dada por Andrade (2000): a UML é uma linguagem de modelagem, é o fundamento a partir do qual se integram linguagens de programação,

ferramentas e processos de desenvolvimento. Ela fornece o conteúdo semântico que formaliza e orienta todo o processo de desenvolvimento. Neste sentido, a UML não é um processo, nem uma ferramenta, nem uma linguagem de programação. Seu objetivo principal é gerar a modelagem dos objetos do mundo real garantindo a interoperabilidade dos recursos envolvidos no processo de desenvolvimento. 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.

3.3 OBJETIVOS

Segundo Eyng (2000) a UML versão 1.3 possui os seguintes objetivos:

a) prover os usuários com uma linguagem de modelagem visual, pronta para uso, para o desenvolvimento de modelos significativos;

b) fornecer mecanismos de ampliação e especialização que possam estender os conceitos principais;

c) ser independente de linguagens de programação e processos de desenvolvimento;

d) prover uma base formal para entendimento da linguagem de modelagem;

e) incentivar o crescimento do mercado de ferramentas orientadas a objeto;

f) suportar conceitos de desenvolvimento de alto nível, tais como colaboração, frameworks, padrões e componentes.

3.4 DIAGRAMAS

Para suprir o seu propósito geral, que é de ser uma linguagem de modelagem gráfica para desenvolvimento orientado a objetos (para as fases de análise e projeto), a UML disponibiliza uma série de diagramas para que seja possível levantar a especificação dos requisitos, a estrutura estática, a parte dinâmica ou comportamental e os aspectos de implementação do sistema ou artefato do sistema (Nau, 1998).

Segue a relação dos diagramas disponibilizados pela UML, com uma breve descrição apresentada por Furlan (1998):

a) diagrama de *Use-Case*: mostra os relacionamentos entre os atores e casos de uso em interação com um sistema;

b) diagrama de Classes: mostra uma coleção de elementos declarativos (estáticos) de modelo, como classes, tipos e seus conteúdos e relações;

c) diagrama de Estado: mostra as transições de estado de uma máquina de estado finita;

d) diagrama de Seqüência: mostra interações de objeto organizados em uma seqüência de tempo, mostrando os objetos que participam na interação e a seqüência de mensagens trocadas;

e) diagrama de Colaboração: mostra interações de objeto, ou seja, as relações entre os objetos. Diagramas de seqüência e de colaboração expressam informações semelhantes, mas as mostram de maneiras diferentes;

f) diagrama de Atividade: é um caso especial de diagrama de estado no qual todos ou a maioria dos estados são estados de ações e a maioria das transições são ativadas por conclusão de ações nos estados de origem;

g) diagrama de Componente: mostra as organizações e dependências entre componentes;

h) diagrama de Execução: representam a disposição dos componentes de hardware que compõem o sistema.

Serão apresentados a seguir alguns diagramas utilizados no trabalho. Todos estes diagramas foram elaborados a partir da ferramenta CASE *System Architect*.

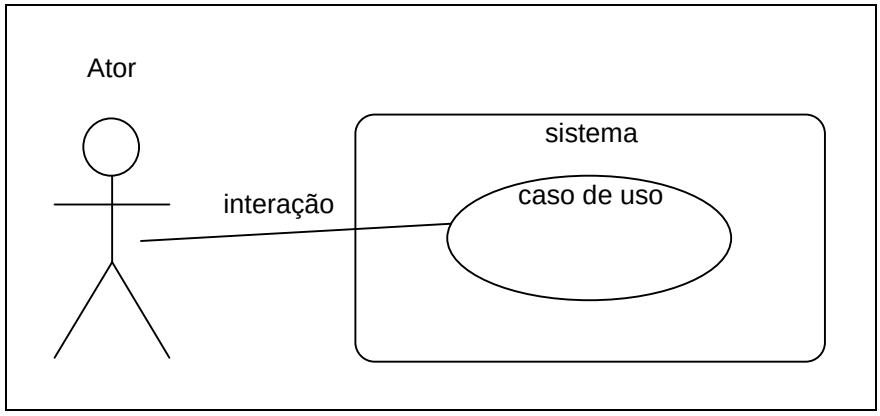
3.4.1 DIAGRAMA DE *USE CASE*

O diagrama de U*se Case* ou também conhecido como diagrama de Caso de Uso fornece um modo de descrever a visão externa do sistema e suas interações com o mundo exterior,

representando um alto nível de funcionalidade intencional mediante o recebimento de um tipo de requisição de usuário (Furlan, 1998).

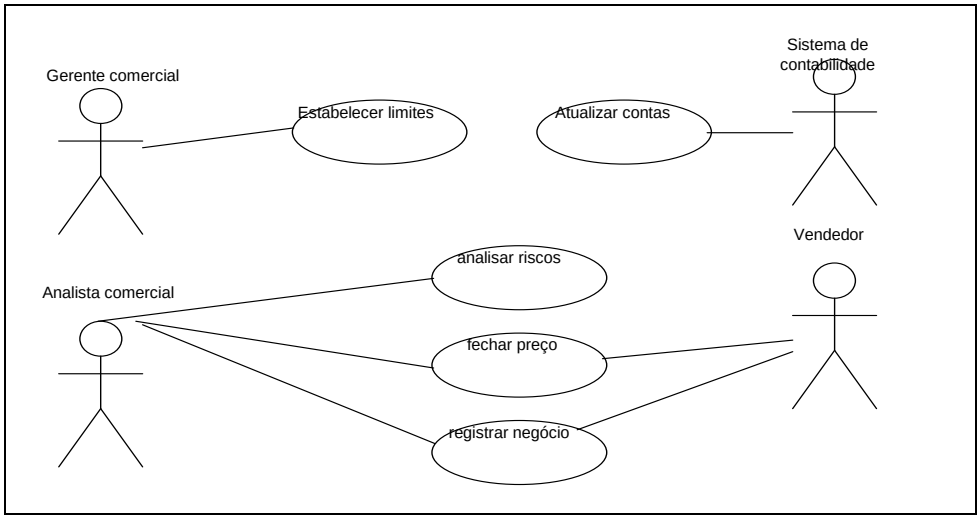
Na modelagem de casos de uso, o sistema é visto como uma caixa-preta que fornece situações de aplicação (ou casos de uso). Não é importante nesse momento compreender como o sistema implementa os casos de uso ou como ocorre o funcionamento interno. Esse diagrama contêm elementos de modelo para o sistema (os atores e os casos de uso) e mostram diferentes relacionamentos, tais como generalização, associação e a dependência entre os elementos. Há quatro elementos básicos em um diagrama de caso de uso: ator, caso de uso, interação e sistema; exemplificados na figura 3.1.

FIGURA 3.1: Diagrama de caso de uso



A figura 3.2 mostra um exemplo de diagrama de caso de uso.

FIGURA 3.2: Exemplo de um diagrama de caso de uso



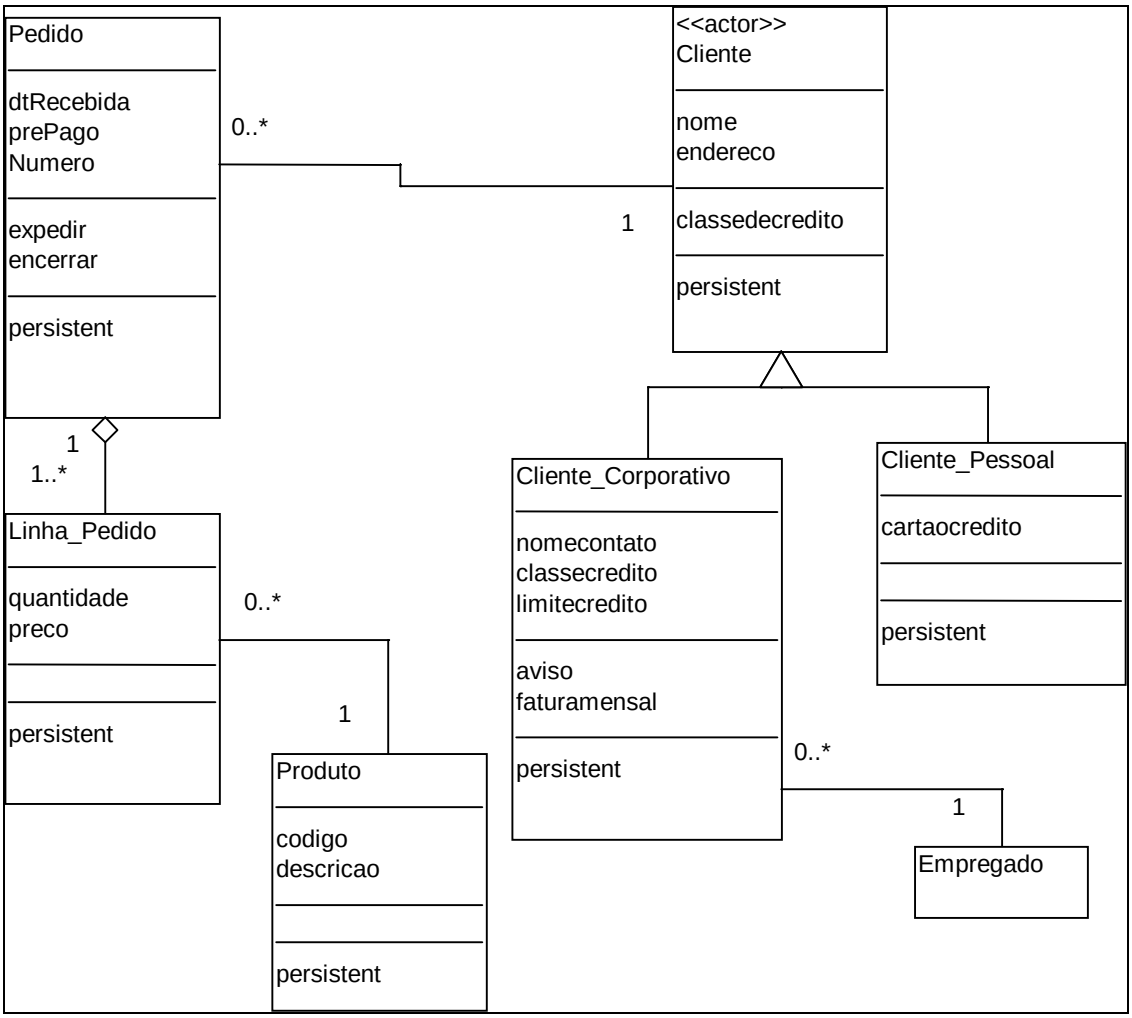
3.4.2 DIAGRAMA DE CLASSES

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), ver relacionamentos na figura 3.3. Todos estes relacionamentos são mostrados no diagrama de classes juntamente com as suas estruturas internas, que são os atributos e operações. O diagrama de classes é considerado estático já que a estrutura descrita é sempre válida em qualquer ponto do ciclo de vida do sistema (Barros, 1998).

Uma outra definição é dada por Furlan (1998). O diagrama de classes é a essência da UML, trata-se de uma estrutura lógica estática em uma superfície de duas dimensões mostrando uma coleção de elementos declarativos de modelo, como classes, tipos e seus respectivos conteúdos e relações.

Há quatro tipos principais de relacionamentos no diagrama de classes: generalização / especialização, agregação, associação e dependência, que foram analisados detalhadamente no capítulo sobre conceitos básicos de orientação a objetos.

FIGURA 3.3: Exemplo de um diagrama de classes



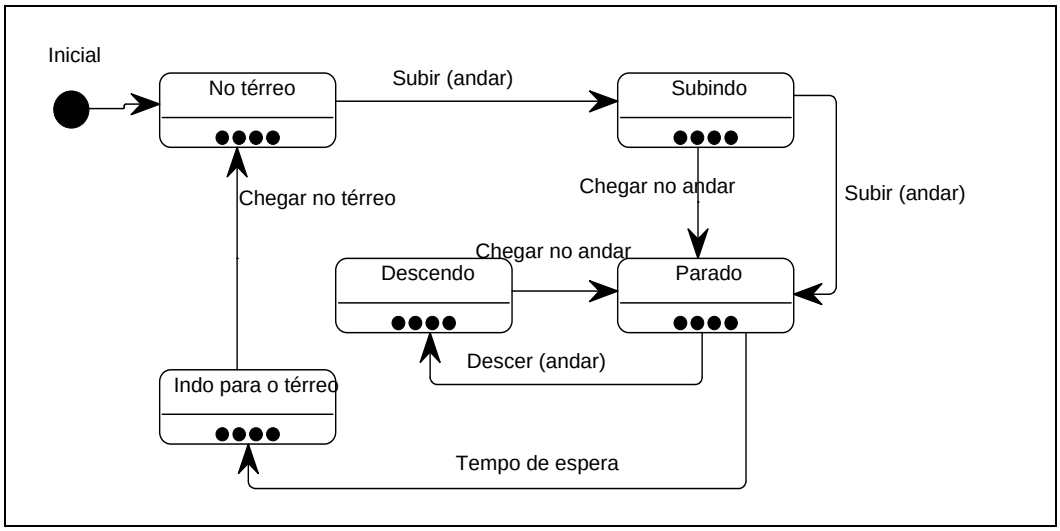
3.4.3 DIAGRAMA DE ESTADO

O diagrama de estado é tipicamente um complemento para a descrição das classes. Este diagrama mostra todos os estados possíveis que objetos de uma certa classe podem se encontrar e mostra também quais são os eventos do sistema que provocam tais mudanças. Os diagramas de estado não são escritos para todas as classes de um sistema, mas apenas para aquelas que possuem um número definido de estados conhecidos e onde o comportamento das classes é afetado e modificado pelos diferentes estados (Barros, 1998).

Diagramas de estado capturam o ciclo de vida dos objetos, subsistemas e sistemas. Eles mostram os estados que um objeto pode possuir e como os eventos (mensagens recebidas, timer, erros, e condições sendo satisfeitas) afetam estes estados ao passar do tempo.

Diagramas de estado possuem um ponto de início e vários pontos de finalização. Um ponto de início (estado inicial) é mostrado como um círculo todo preenchido, e um ponto de finalização (estado final) é mostrado como um círculo em volta de um outro círculo menor preenchido. Um estado é mostrado como um retângulo com cantos arredondados. Entre os estados estão as transições, mostradas como uma linha com uma seta no final de um dos estados. A transição pode ser nomeada com o seu evento causador. Quando o evento acontece, a transição de um estado para outro é executada ou disparada (ver figura 3.4).

FIGURA 3.4: Diagrama de estado

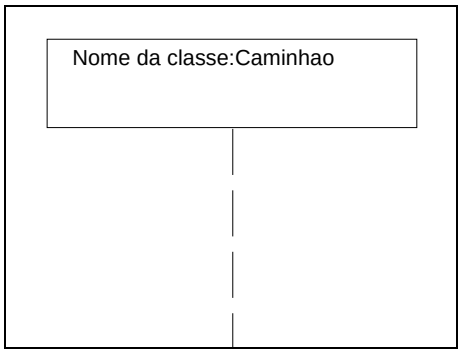


Conforme Barros (1998), uma transição de estado normalmente possui um evento ligado a ela. Se um evento é anexado a uma transição, esta será executada quando o evento ocorrer. Se uma transição não possuir um evento ligado a ela, a mesma ocorrerá quando a ação interna do código do estado for executada (se existir ações internas como entrar, sair, fazer ou outras ações definidas pelo desenvolvedor). Então quando todas as ações forem executadas pelo estado, a transição será disparada e serão iniciadas as atividades do próximo estado no diagrama de estados.

3.4.4 DIAGRAMA DE SEQÜÊNCIA

Segundo Prebianca (1998), um diagrama de seqüência apresenta as interações entre objetos de um ponto de vista temporal, enfocado na transmissão de mensagens seqüenciais. Um objeto é representado por um retângulo e uma barra vertical chamada de linha da vida do objeto (conforme figura 3.5).

FIGURA 3.5 : Representação de uma classe no diagrama de seqüência



Os objetos comunicam-se trocando mensagens, representadas por setas horizontais saindo do transmissor ao receptor da mensagem. A ordem da mensagem enviada é indicada pela posição da mensagem no eixo vertical (Prebianca, 1998).

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. O diagrama de seqüência consiste em um número de objetos mostrados em linhas verticais. O decorrer do tempo é visualizado observando-se o diagrama no sentido vertical de cima para baixo. As mensagens enviadas por cada objeto são simbolizadas por setas entre os objetos que se relacionam (Barros, 1998).

Diagramas de seqüência possuem dois eixos:

- eixo vertical: que mostra o tempo;

- eixo horizontal: que mostra os objetos envolvidos na seqüência de uma certa atividade.

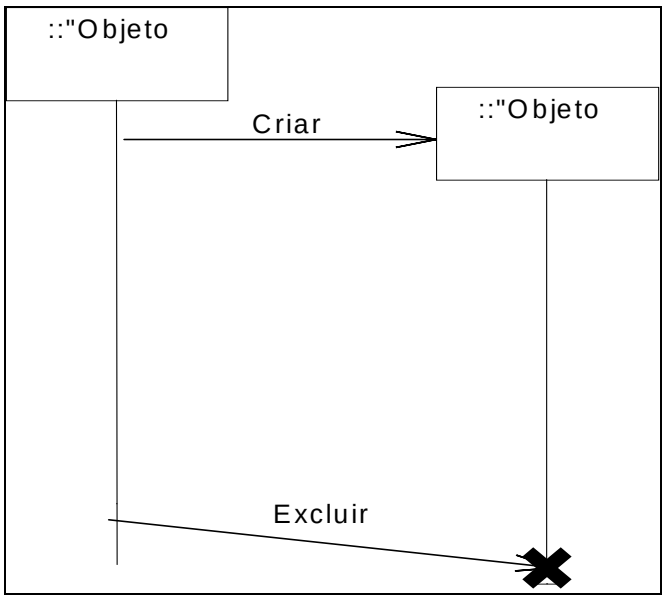
Eles também mostram as interações para um cenário específico de uma certa atividade do sistema.

No eixo horizontal estão os objetos envolvidos na seqüência. Cada um é representado por um retângulo de objeto e uma linha vertical pontilhada chamada de linha de vida do objeto, indicando a execução do objeto durante a seqüência.

A comunicação entre os objetos é representada como linha com setas horizontais simbolizando as mensagens entre as linhas de vida dos objetos. A seta especifica se a mensagem é síncrona, assíncrona ou simples. As mensagens podem possuir também números seqüenciais, eles são utilizados para tornar mais explícito as seqüências no diagrama.

Os diagramas de seqüência podem mostrar objetos que são criados ou destruídos como parte do cenário documentado pelo diagrama. Um objeto pode criar outros objetos através de mensagens. A mensagem que cria ou destrói um objeto é geralmente síncrona, representada por uma seta sólida, observe na figura 3.6.

FIGURA 3.6 : Exemplo de um digrama de seqüência que cria objetos e destrói



3.4.5 DIAGRAMA DE ATIVIDADE

Um diagrama de atividade é organizado de acordo com ações e principalmente, focado para representar o comportamento interno de um método (a implementação de uma operação) ou um caso de uso (Prebianca, 1998).

Segundo Furlan (1998), um diagrama de atividade representa o estado de execução do método, em outras palavras, o estado do objeto que executa o método. Sob uma perspectiva conceitual, uma atividade é alguma tarefa que precisa ser feita, independentemente se for por um computador ou uma pessoa; e através de uma perspectiva de implementação, uma atividade é um método sobre uma classe. Seu propósito é entender o algoritmo envolvido na execução de um método.

3.4.5.1 REPRESENTAÇÃO DAS ATIVIDADES

Tanto o diagrama de estado como o diagrama de atividade representa uma seqüência de passos, no diagrama de atividade o estado da execução de um mecanismo é representado com uma seqüência de passos agrupados consecutivamente. Porém, dada à natureza da implementação das operações, na qual a maioria dos eventos, simplesmente corresponde ao final da atividade precedente, é aconselhável ter uma representação simplificada para apresentar as atividades. Neste contexto, uma atividade é representada como um estereótipo de um estado. Não é necessário distinguir estados, atividades e eventos, sistematicamente (Prebianca, 1998). Assim sendo, uma atividade é representada por um retângulo arredondado, da mesma maneira como os estados, só que mais arredondado.

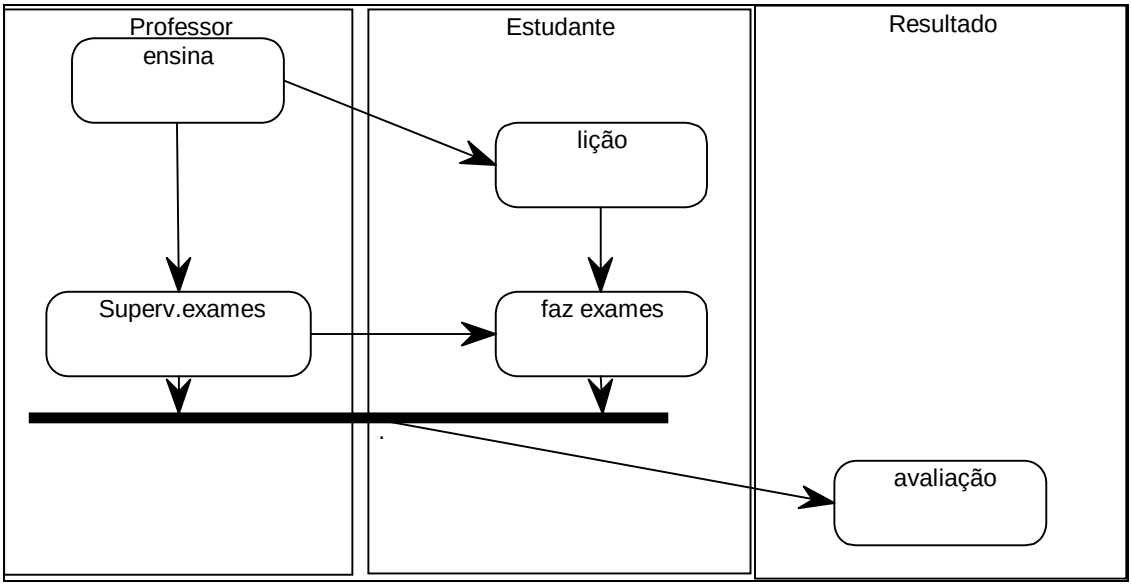
Cada atividade representa um estado particular dentro da execução de um método encapsulado. São unidas por transições automáticas, representadas por setas, da mesma maneira como transições dentro do diagrama de estados. Quando uma atividade termina, a transição é ativada e a próxima atividade começa. As atividades não possuem transições internas ou transições ativadas por eventos

Freqüentemente, segundo Furlan (1998), o mesmo objeto é manipulado por várias atividades sucessivas sendo possível desenhar setas para todas atividades pertinentes. Para maior clareza, no entanto, o objeto é exibido várias vezes em um diagrama e cada aparecimento denota um ponto diferente em sua vida. Para distinguir os vários aparecimentos

do mesmo objeto, o estado do objeto a cada ponto pode ser colocado entre colchetes e juntado ao nome do objeto, por exemplo, Pedido de compra [aprovado].

No diagrama de atividade ocorrem sincronizações entre fluxos de controle usando barras de sincronização. Uma barra de sincronização torna possível abrir e fechar partes paralelas dentro do fluxo de execução de um método ou de um caso de uso. As transições que pertencem ao começo de uma barra de sincronização são simultaneamente ativadas.

FIGURA 3.7: Exemplo de diagrama de atividade



4 DESENVOLVIMENTO DO PROTÓTIPO

Este capítulo apresenta a especificação do protótipo que foi construído a partir do uso de alguns diagramas da UML na ferramenta CASE *System Architect*. Esta experiência foi importante para o processo posterior de avaliação da qualidade da ferramenta CASE.

4.1 DEFINIÇÃO DO PROTÓTIPO

O protótipo desenvolvido auxiliará na avaliação dos cursos de graduação da área de Computação (Bacharelado em Ciências da Computação, Engenharia de Computação, Bacharelado em Sistemas de Informação e Licenciatura em Computação).

Para esta avaliação são usados indicadores de qualidade e seus padrões, definidos pela Comissão de Especialistas de Ensino de Informática do MEC/SESU (MEC/SESU, 1997). Esses padrões de qualidade deverão ser cumpridos pelos cursos da área, como condição para serem recomendados. Para a avaliação dos cursos, as Instituições de ensino deverão preencher o formulário de avaliação, integrado a esses padrões de qualidade.

Cada indicador de qualidade contém, quando foi possível quantificá-lo, seus padrões de qualidade e, adicionalmente, para efeito de avaliação e verificação:

a) um formulário a ser preenchido pela Instituição de ensino;

b) um formulário de avaliação a ser preenchido pela Comissão de Avaliação do MEC.

Primeiramente é feito um relatório de pedido de avaliação que é enviado para o MEC. Após o aceite da avaliação pelo MEC será feito o cadastramento:

a) de todos os softwares, livros textos e livros referências, de todas as disciplinas, da grade curricular dos cursos, dos dados dos cursos oferecidos, entre outros;

b) de todo o corpo docente incluindo dados curriculares, informações de disciplinas já lecionadas pelo docente, carga horária de cada docente, dados completos do coordenador de cada curso, principais referências da produção científica, entre outros;

c) de toda a infra-estrutura, incluindo informações da Biblioteca, do espaço físico de toda a Instituição, relação de todos os livros por disciplina, cadastro de todos os equipamentos dos laboratórios, um plano de manutenção e atualização tecnológica dos laboratórios;

d) de todo plano pedagógico incluindo toda a estrutura curricular, relação de softwares, livros textos e livros referências para cada disciplina, entre outros;

e) do desempenho do curso incluindo várias descrições de critérios essenciais do curso na Instituição, por exemplo, conceitos do exame nacional de cursos, perfil dos egressos, metodologia do curso em função do perfil do egresso, apresentar critérios de desligamentos dos alunos, indicar o relacionamento estrutural da coordenação e colegiado do curso com os órgãos mais próximos da estrutura, entre outros.

Através de padrões de qualidade, definidos pelo MEC, obtêm-se um conceito. São avaliados os seguintes capítulos: corpo docente, plano pedagógico, infra-estrutura e desempenho do curso; em cada um destes, o protótipo calcula o conceito, e a comissão pode alterar este conceito previamente calculado baseado nos padrões de qualidade, justificando através de um descritivo, descrevendo ainda pontos fracos e pontos fortes. O resultado final é um relatório com o resultado da avaliação, separado por capítulo, mostrando o indicador de qualidade e o conceito obtido com a devida justificativa do conceito, esta feita pela comissão. O protótipo gera este relatório final e a comissão poderá alterar os conceitos.

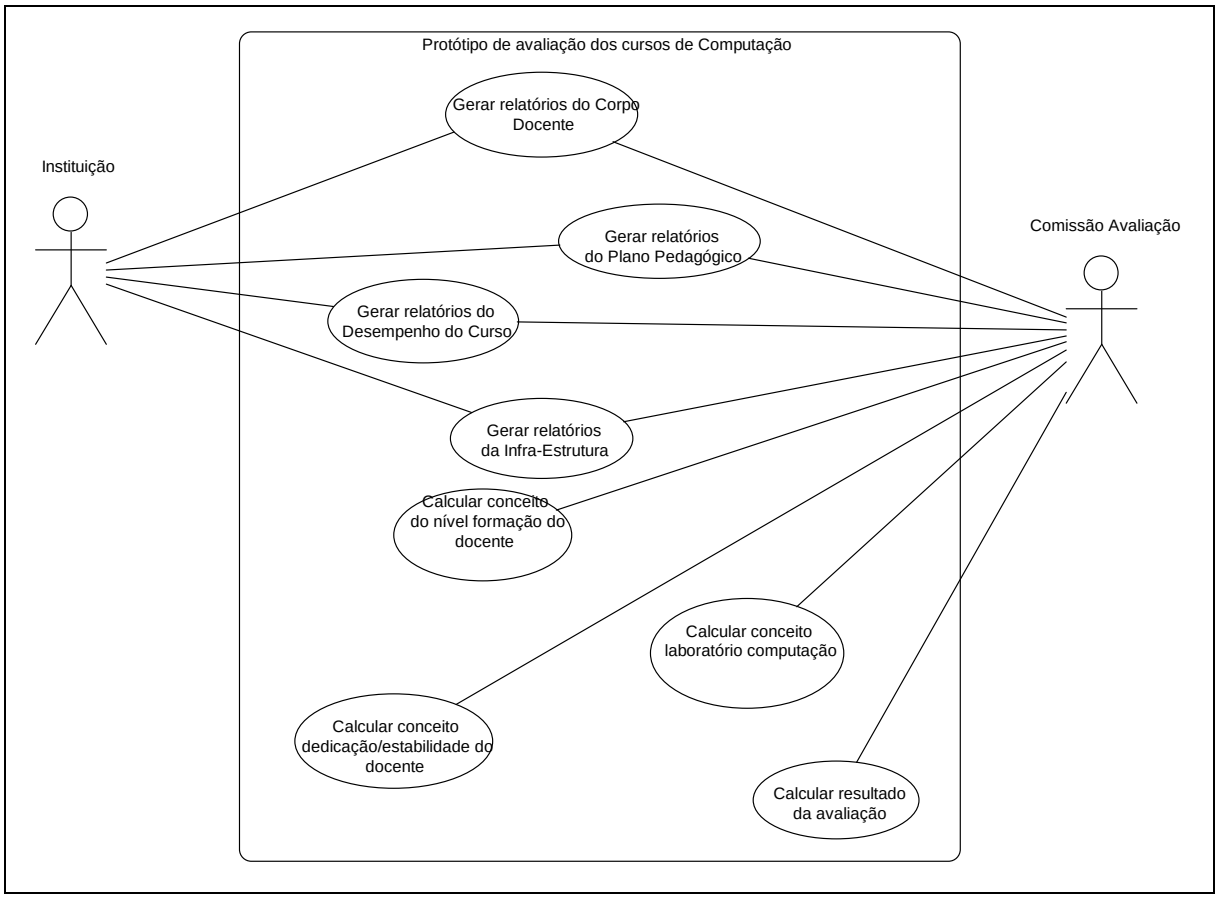
O ambiente escolhido para o desenvolvimento do protótipo foi definido levando-se em consideração o conhecimento da autora na linguagem de programação Delphi.

4.2 DIAGRAMAS

Neste tópico serão apresentados alguns diagramas da UML. Foram utilizados o diagrama de Caso de Uso, o diagrama de Classes, o diagrama de Seqüência, o diagrama de Colaboração e o diagrama de Atividade. Estes diagramas possibilitam uma visão geral do funcionamento do protótipo, utilizando a representação gráfica.

Na figura 4.1 é mostrado o diagrama de Caso de Uso do protótipo. Neste diagrama tem-se uma visão do que o protótipo deve fazer, sem detalhamento.

FIGURA 4.1: Diagrama de Caso de Uso do protótipo



A seguir apresenta-se a especificação textual de cada caso de uso identificado no diagrama da figura 4.1.

- Caso de Uso Gerar relatórios do corpo docente

Para avaliar o corpo docente são necessários vários relatórios como a Ficha de cada docente, relação das disciplinas oferecidas nos últimos cinco anos, referências publicadas pelos docentes, resumo dos docentes, estabilidade do corpo docente, tabela de carga horária/dedicação ao aluno, tabela resumo trabalho dos docentes, descrição da política aperfeiçoamento/qualificação/atualização dos docentes, descrição do coordenador do curso.

- Caso de Uso Gerar relatórios da infra-estrutura

Para avaliar a infra-estrutura são necessários vários relatórios como a tabela com a quantidade livros/disciplina, descrição da Biblioteca, quadro do uso efetivo dos

alunos/equipamentos, listagem dos cursos que usam laboratório, quadro com equipamentos do laboratório, quadro com as licenças de software, descrição da disponibilidade do uso equipamentos/alunos, descrição do plano de manutenção, descrição do plano de atualização tecnológica, quadro dos parâmetros do laboratório hardware.

- Caso de Uso Gerar relatórios do plano pedagógico

Para avaliar o plano pedagógico são necessários vários relatórios como a descrição do perfil do egresso, descrição da metodologia do curso, currículo do curso com as diretrizes, descrição da obtenção do grau, quadro disciplinas com livros/softwares, descrição critério jubilamento, descrição critério reprovações, descrição desligamento, descrição pesquisa/extensão em Computação, descrição da administração do curso, informações participação estudantes órgãos colegiados, informação estatuária de centro acadêmico.

- Caso de Uso Gerar relatórios do desempenho do curso

Para avaliar o desempenho do curso são necessários vários relatórios como os conceitos do exame nacional de cursos e o quadro de desempenho do curso.

- Caso de Uso Calcular conceito do nível formação do docente

Para calcular este conceito é necessário ter o percentual de doutores, mestres, especialistas na área Computação, não especialistas na área de Computação, todos estes percentuais devem ser separados para docentes que lecionam para cursos com atividade fim ou meio. Após, é definido o conceito conforme um padrão de qualidade, por exemplo, para atribuir o conceito A nos cursos de atividade fim os percentuais de doutores e mestres deve ser igual ou superior a 90% e o percentual de doutores ser superior ou igual a 50%.

- Caso de Uso Calcular conceito dedicação/estabilidade do docente

Existem padrões de qualidade para definir o conceito de dedicação e de estabilidade do docente. O padrão para a estabilidade é o percentual de docentes com estabilidade em relação ao total de docentes do curso. O padrão de dedicação do docente é o percentual de trabalho com regime integral para docentes na área de Computação e de outras áreas, ainda separados por cursos com atividade fim ou meio.

- Caso de Uso Calcular conceito laboratório computação

Existe um padrão de qualidade que é uma tabela de conceitos conforme um percentual de alunos por posto de trabalho. Baseado neste padrão calcula-se o percentual de alunos por posto de trabalho e consulta-se na tabela para obter o conceito.

- Caso de Uso Calcular resultado da avaliação

É definido um conceito global para cada capítulo, observando-se regras de exceção. Se for autorização o conceito global do corpo docente, por exemplo, será convertido para D ou E e, se for reconhecimento, para E. Para os quatro capítulos existem uma regra de exceção. Após, define-se o conceito global do curso, também conforme o objetivo for autorização ou reconhecimento. Se for primeira avaliação e o objetivo for autorização e o conceito global dos capítulos corpo docente, plano pedagógico e infra-estrutura recebeu o conceito global D ou E, então o conceito global do curso será DILIGÊNCIA. Se for primeira avaliação e o objetivo for reconhecimento e o conceito global dos capítulos corpo docente, plano pedagógico e infra-estrutura recebeu o conceito global E, então o conceito global do curso será DILIGÊNCIA.

Inicialmente foi feito o diagrama entidade-relacionamento visto na figura 4.2 e a partir deste modelo a ferramenta CASE *System Architect* gera o diagrama de classes conforma visto na figura 4.3.

FIGURA 4.2: Modelo entidade-relacionamento do protótipo

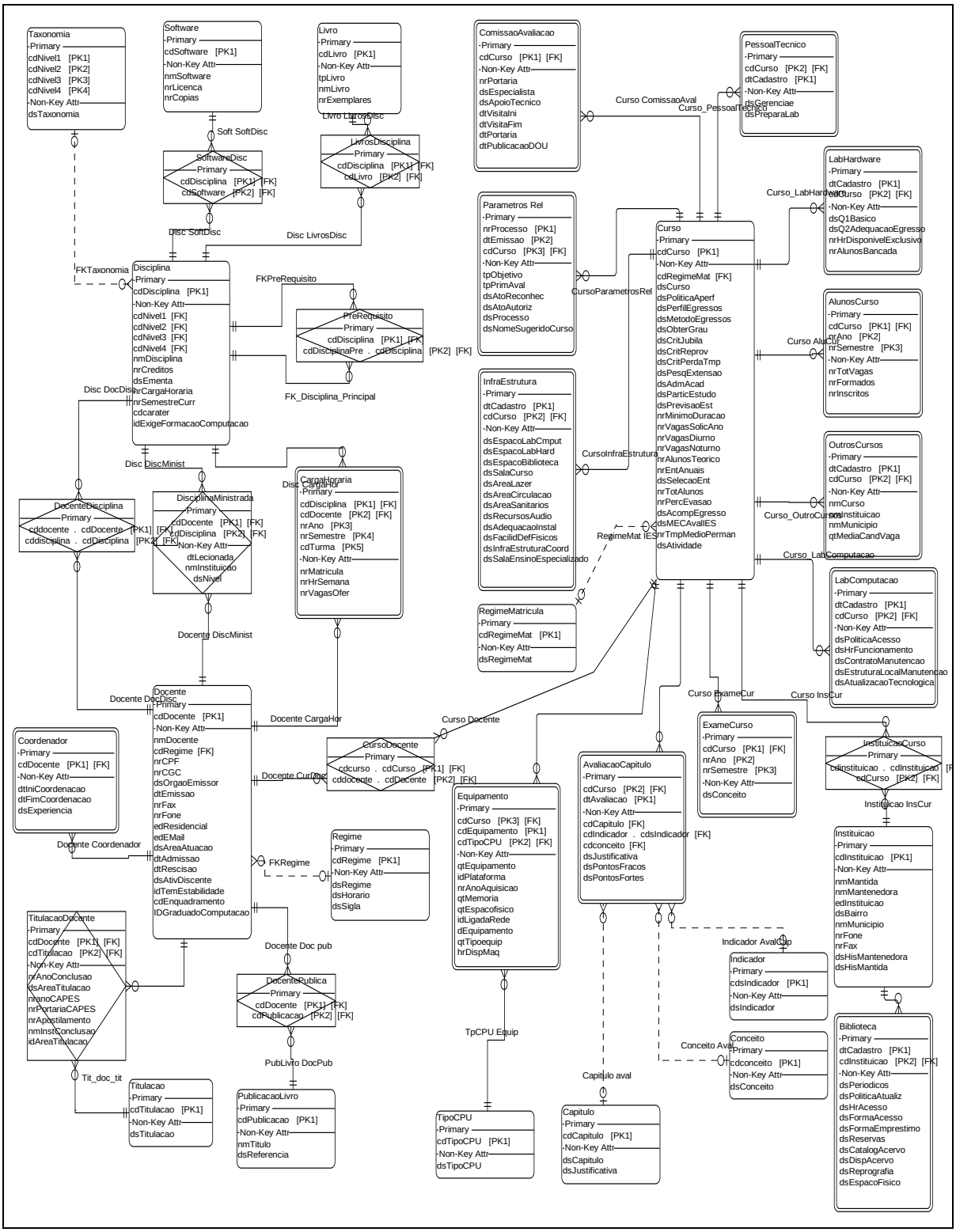
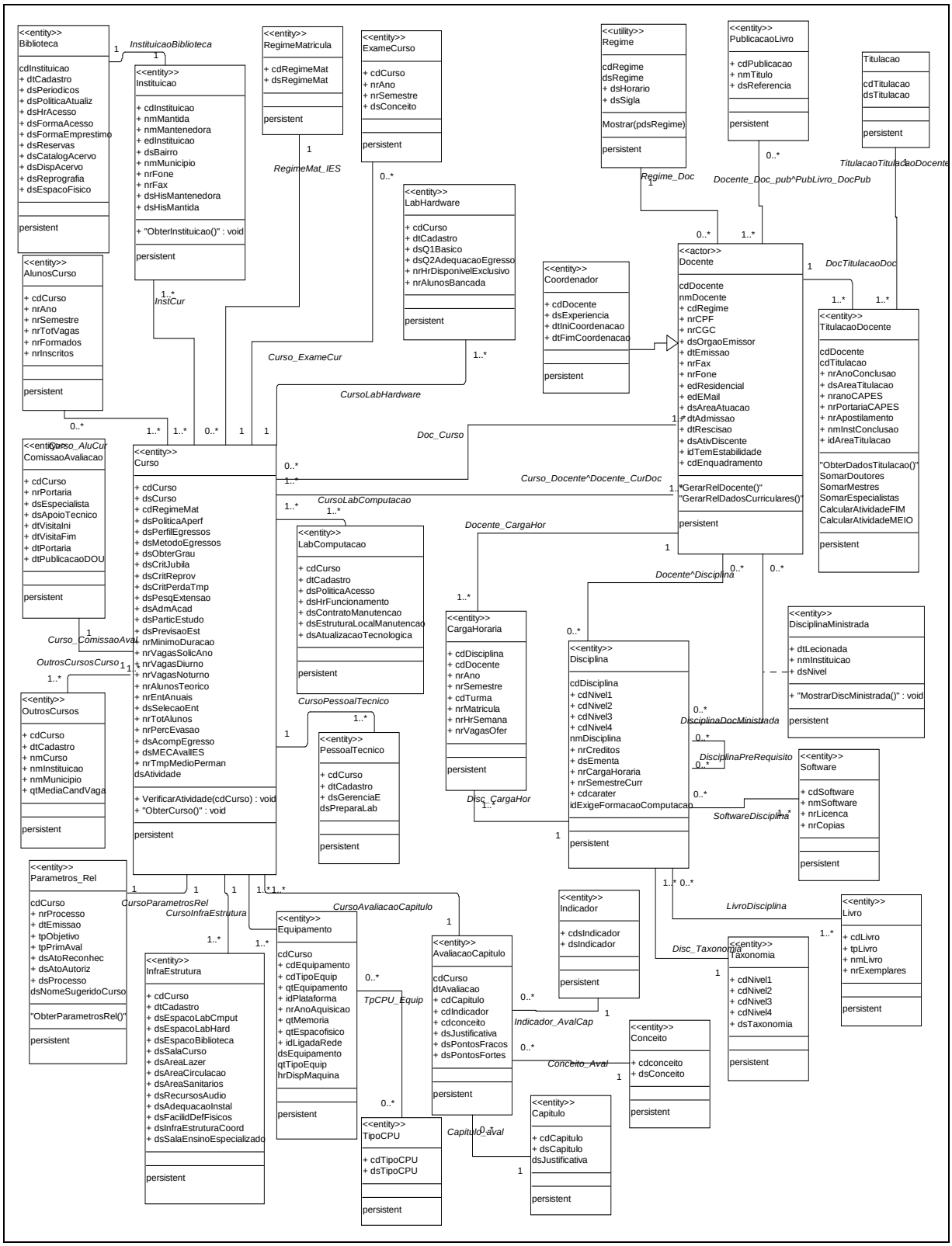
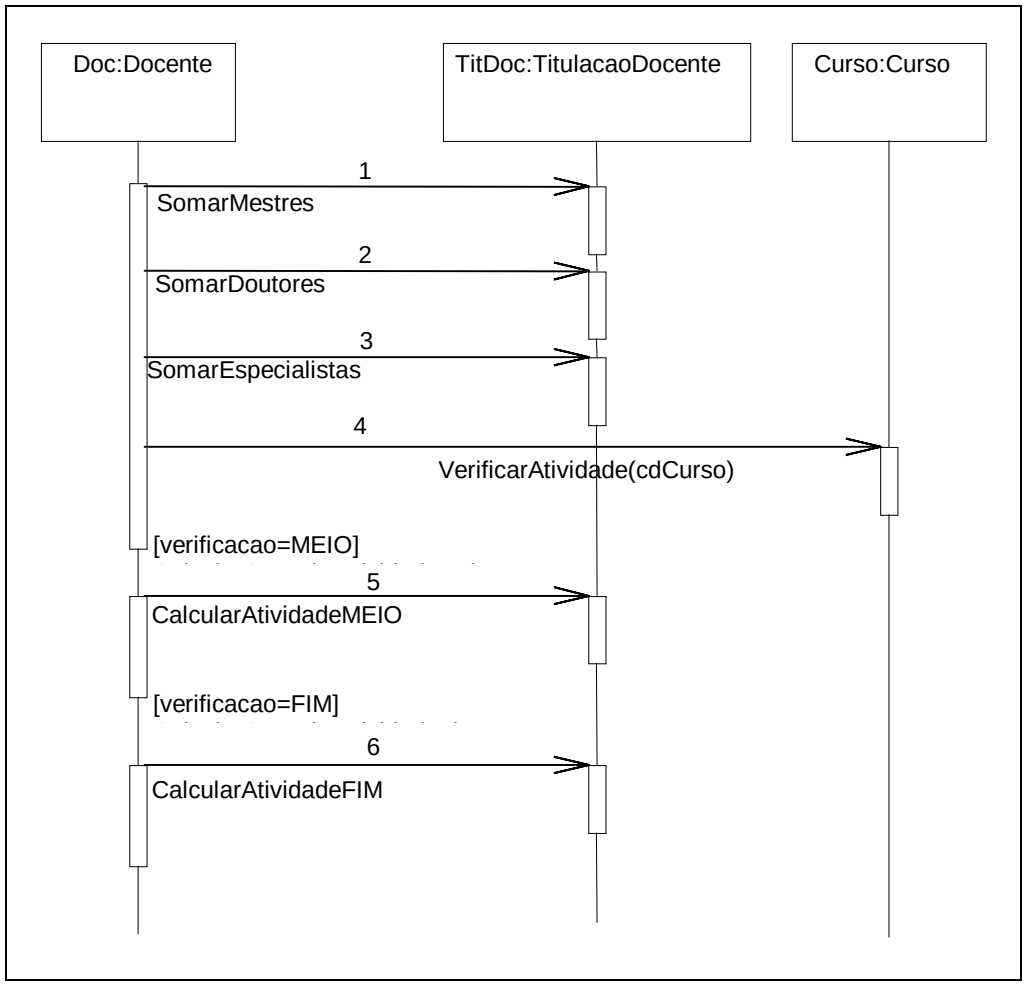


FIGURA 4.3: Diagrama de classes do protótipo



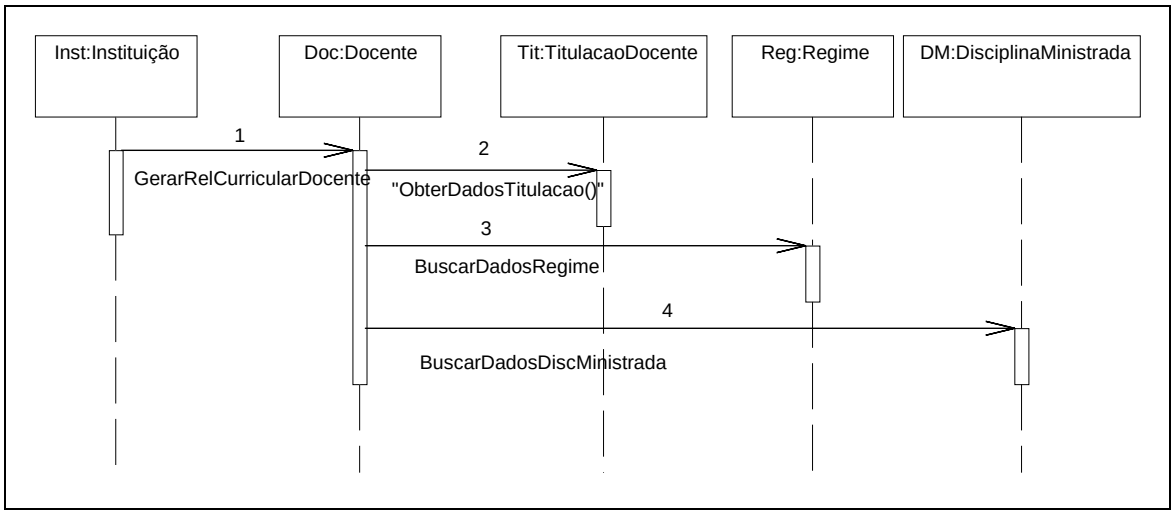
Na figura 4.4 é mostrado o diagrama de seqüência do caso de uso calcular conceito do nível de formação. Este conceito é calculado baseado numa tabela de conceitos, onde o conceito A para cursos com atividade fim é obtido quando o percentual de doutores e o percentual de mestres for maior ou igual a 65% e o percentual de doutores for maior ou igual a 35%. Para calcular os percentuais é necessário somar todos os mestres, doutores e especialistas, após verifica-se a atividade do curso que pode ser ‘meio’ ou ‘fim’ pois para cada atividade o conceito é calculado de forma diferente.

FIGURA 4.4: Diagrama de seqüência ‘CalcularConceitoNivelFormacao’



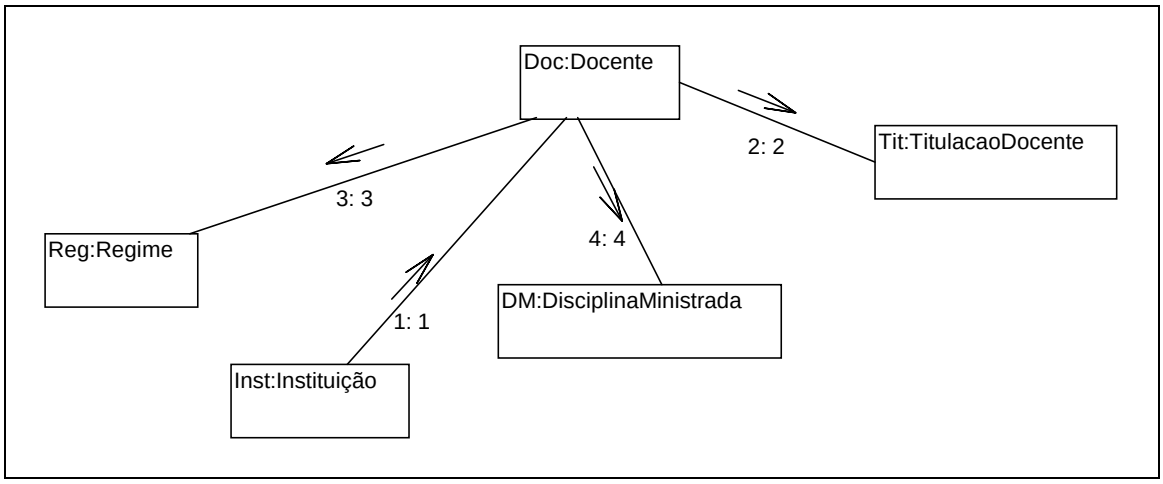
Na figura 4.5 é mostrado o diagrama de seqüência do caso de uso gerar relatório curricular do docente.

FIGURA 4.5: Diagrama de seqüência ‘GerarRelatorioCurricularDocente’



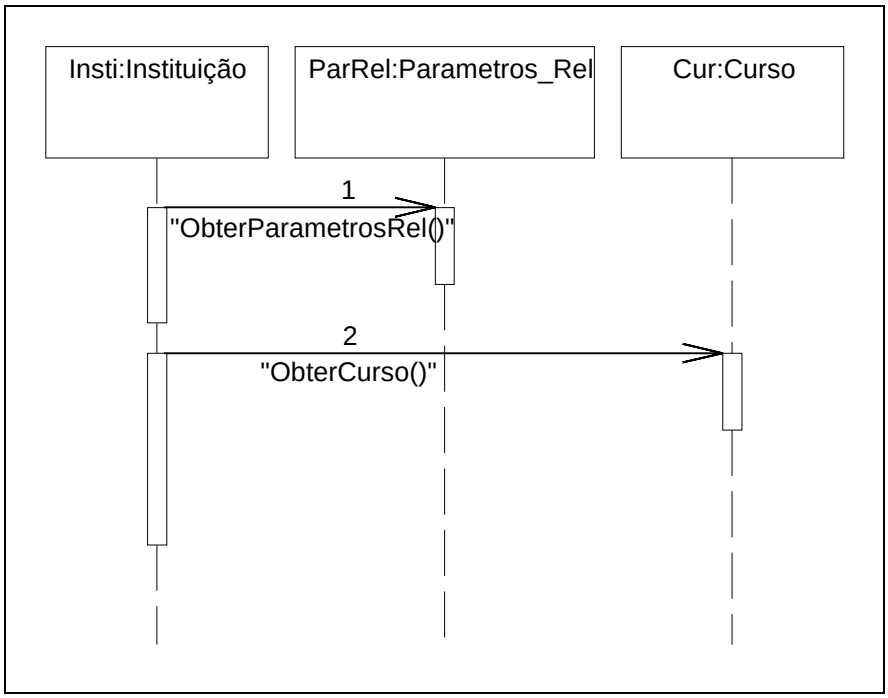
Na figura 4.6 é mostrado o diagrama de colaboração do caso de uso gerar relatório curricular do docente, este tipo de diagrama ilustra a interação espacialmente, sem levar em consideração o tempo.

FIGURA 4.6: Diagrama de colaboração ‘GerarRelatorioCurricularDocente’



Na figura 4.7 é mostrado o diagrama de seqüência do caso de uso gerar relatório de avaliação final, ilustrando uma visão temporal.

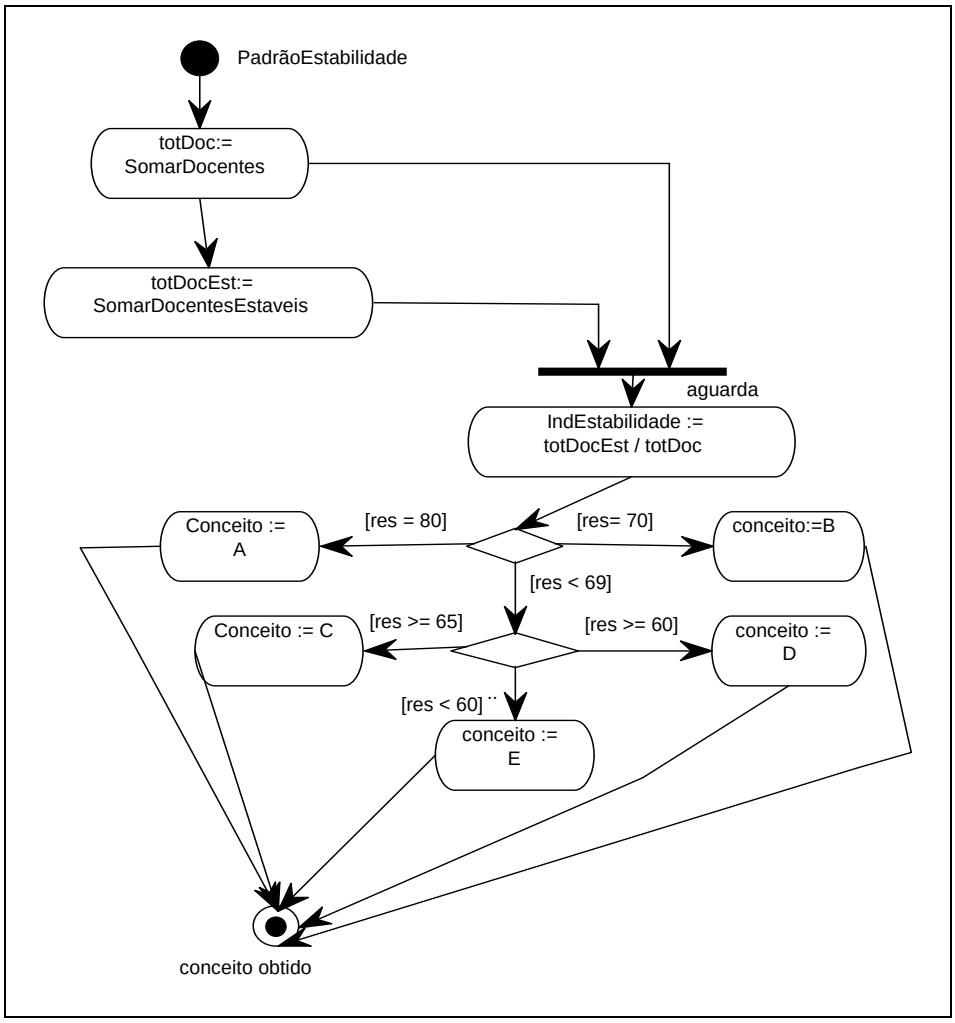
FIGURA 4.7: Diagrama de seqüência ‘GerarRelatorioAvaliacaoFinal’



Na figura 4.8 é mostrado um diagrama de atividade relatando como calcular o conceito para a estabilidade do docente, que é baseado no padrão da qualidade da estabilidade. O índice da estabilidade é mostrado na fórmula seguinte. Para obter o número de professores estáveis é preciso verificar na tabela Docente se professor é estável e para obter o número total de professores soma-se todos os professores da tabela Docente. Calculado o índice verifica-se na tabela de conceitos em qual conceito a instituição se encontra. A tabela de conceitos estabelece que, o conceito A é indicado para o índice de estabilidade maior ou igual a 80%, o conceito B é indicado para o índice de estabilidade maior ou igual a 70%, o conceito C é indicado para o índice de estabilidade maior ou igual a 65%, o conceito D é indicado para o índice de estabilidade maior ou igual a 60% e o conceito E é indicado para o índice de estabilidade menor que 60%.

$$\text{Índice de Estabilidade} = \frac{\text{número de professores estáveis}}{\text{número total de professores}}$$

FIGURA 4.8: Diagrama de atividade do cálculo da estabilidade

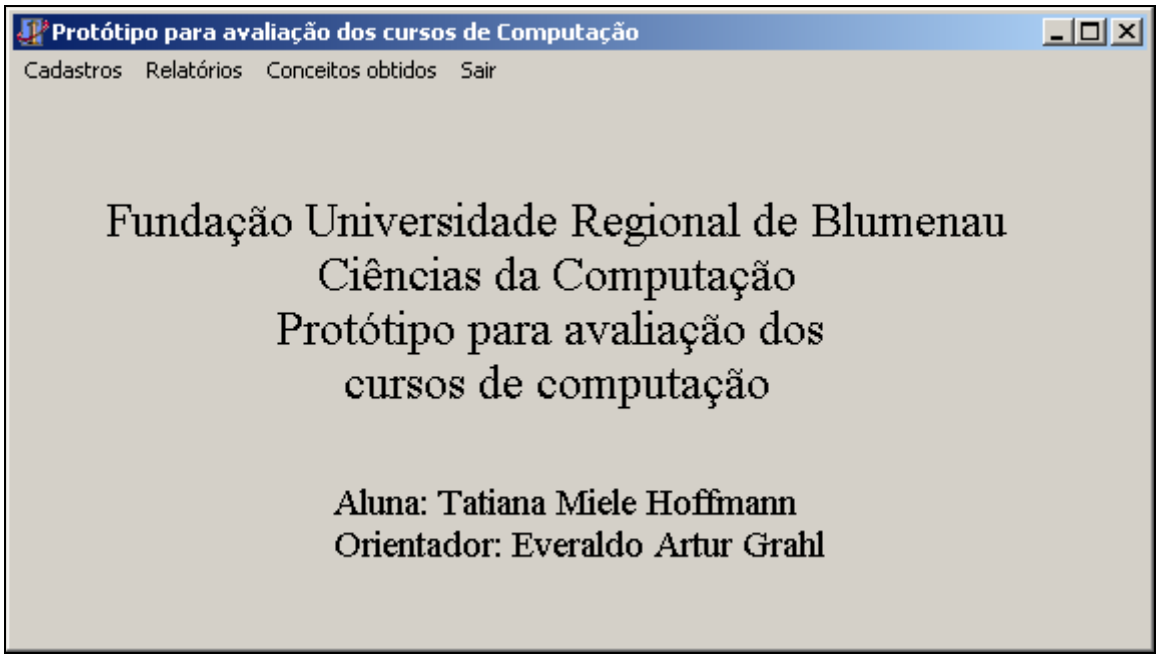


A variável *totdoc* contém o total de docentes do curso, a variável totdocest contém o total de docentes estáveis. O índice da estabilidade está na variável indestabilidade e conforme o resultado do índice, é indicado o conceito.

4.3 ESPEFICICAÇÃO DO PROTÓTIPO

Serão mostradas neste tópico, algumas telas e relatórios principais do protótipo. Na figura 4.9 é ilustrada a tela principal, que possui no menu as opções de cadastros, relatórios e os conceitos obtidos de cada capítulo.

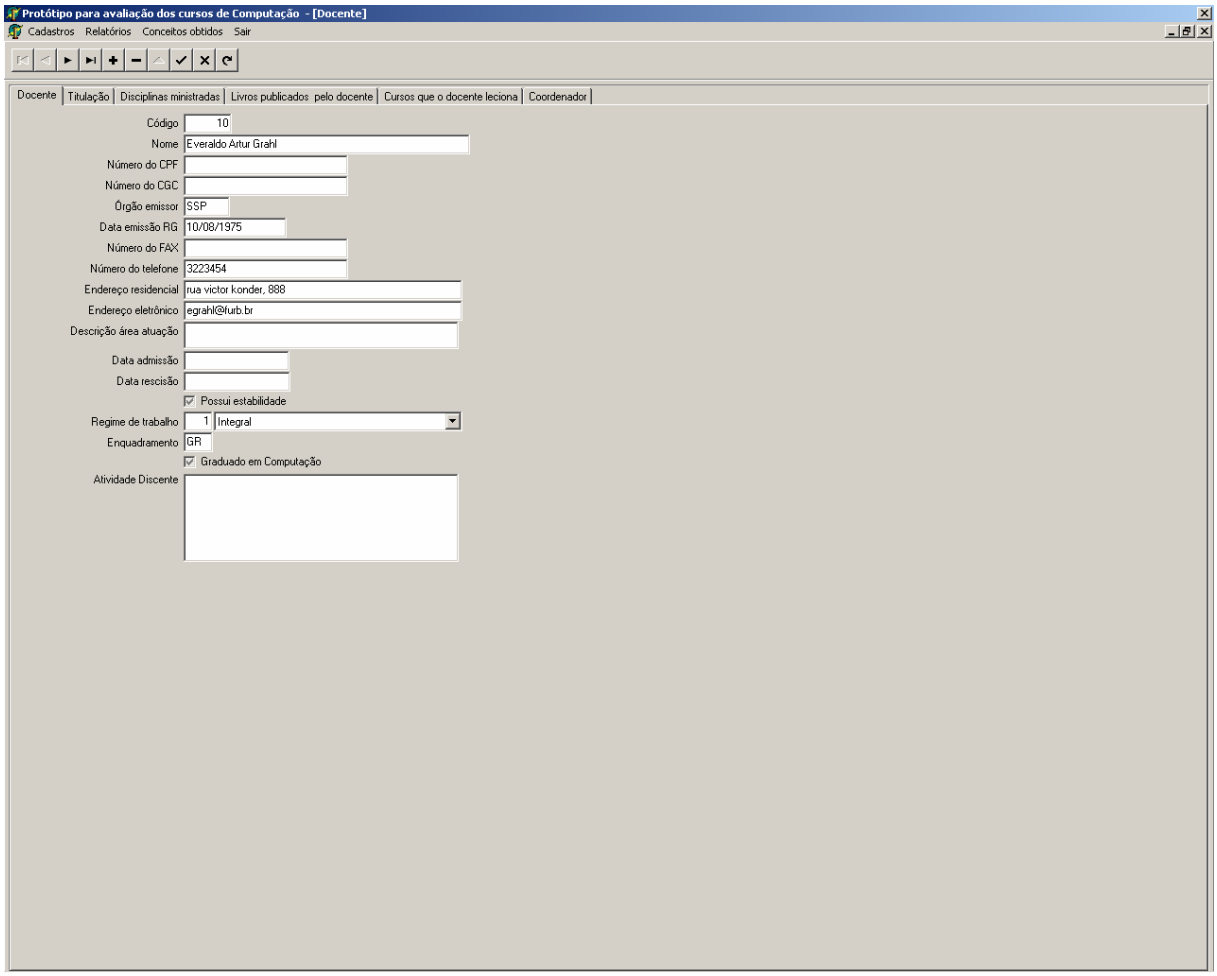
FIGURA 4.9: Tela principal do protótipo



A seguir será mostrado passo a passo o funcionamento para se calcular o indicador de qualidade ‘nível de formação e adequação do corpo docente’ do capítulo do corpo docente.

Primeiramente, é preciso que se faça o cadastro de todos os docentes. A figura 4.10 apresenta a tela de cadastro dos docentes.

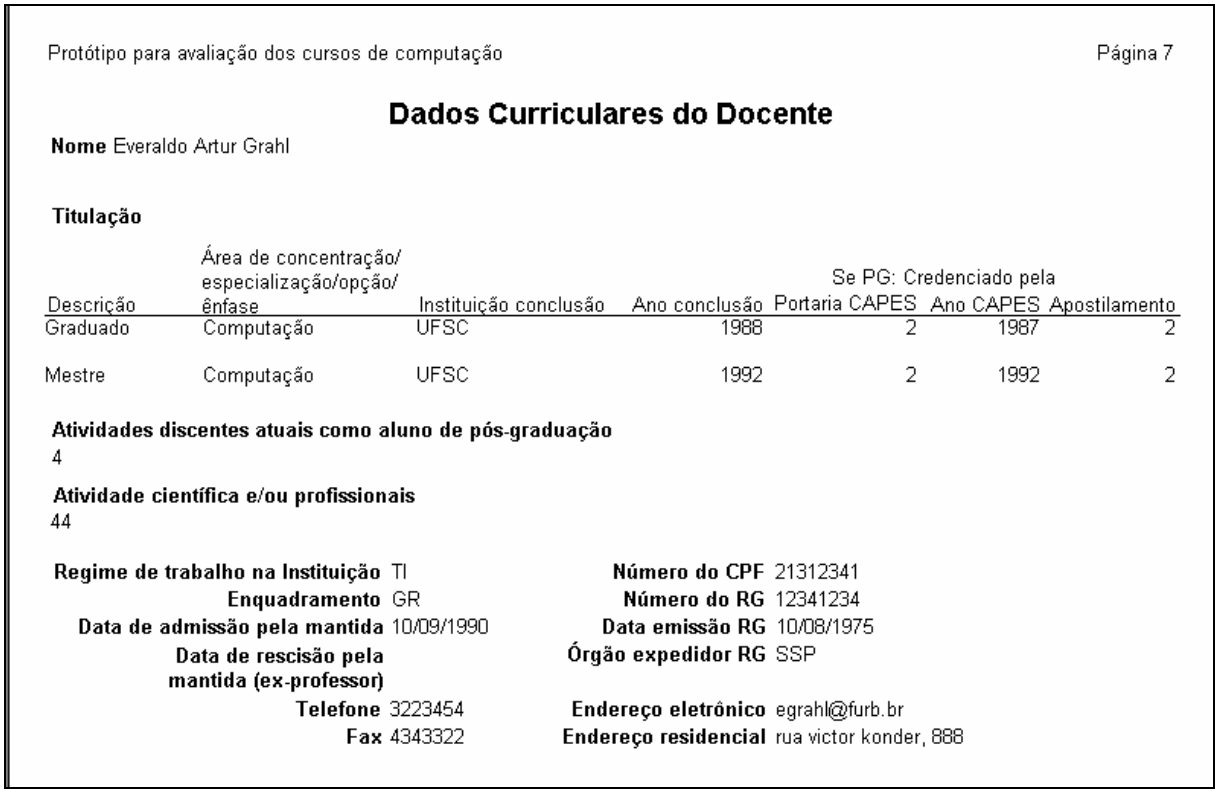
FIGURA 4.10: Cadastro de docentes



Na opção ‘Titulação’ cadastram-se suas titulações. Se o docente for coordenador deve-se cadastrar na opção ‘Coordenador’ mais algumas informações referentes ao seu cargo. Na opção ‘Livros publicados pelo docente’ são cadastrados os livros publicados. Na opção ‘Disciplinas ministradas’ são cadastradas as disciplinas que já foram ministradas pelo docente. Na opção ‘Cursos que o docente leciona’ são cadastrados os cursos em que o docente leciona. Também devem ser cadastrados os tipos de regime de trabalho da instituição.

Para a avaliação deste indicador, devem ser impressos alguns relatórios, que serão apresentados na figura 4.11, 4.12, 4.13 e 4.14. O relatório da figura 4.11 deve ser impresso para cada docente da instituição.

FIGURA 4.11: Relatório dos dados curriculares do docente



Protótipo para avaliação dos cursos de computação Página 7

Dados Curriculares do Docente

Nome Everaldo Artur Grahl

Titulação

Descrição	Área de concentração/ especialização/opção/ ênfase	Instituição conclusão	Ano conclusão	Se PG: Credenciado pela Portaria CAPES	Ano CAPES	Apostilamento
Graduado	Computação	UFSC	1988	2	1987	2
Mestre	Computação	UFSC	1992	2	1992	2

Atividades discentes atuais como aluno de pós-graduação
4

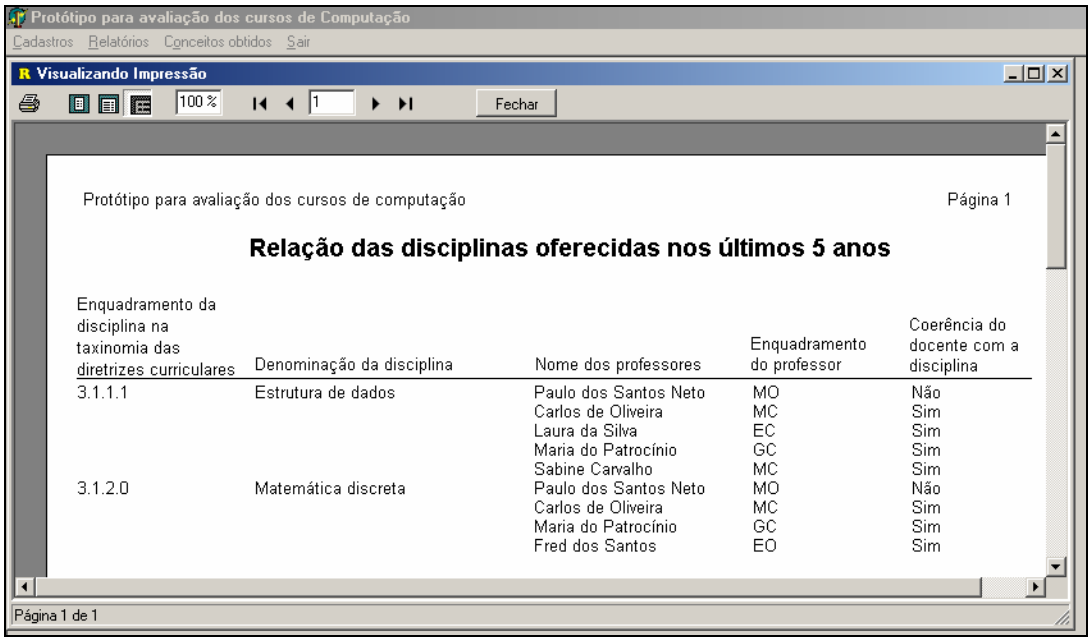
Atividade científica e/ou profissionais
44

Regime de trabalho na Instituição TI
Enquadramento GR
Data de admissão pela mantida 10/09/1990
Data de rescisão pela mantida (ex-professor)
Telefone 3223454
Fax 4343322

Número do CPF 21312341
Número do RG 12341234
Data emissão RG 10/08/1975
Órgão expedidor RG SSP
Endereço eletrônico egrahl@furb.br
Endereço residencial rua victor konder, 888

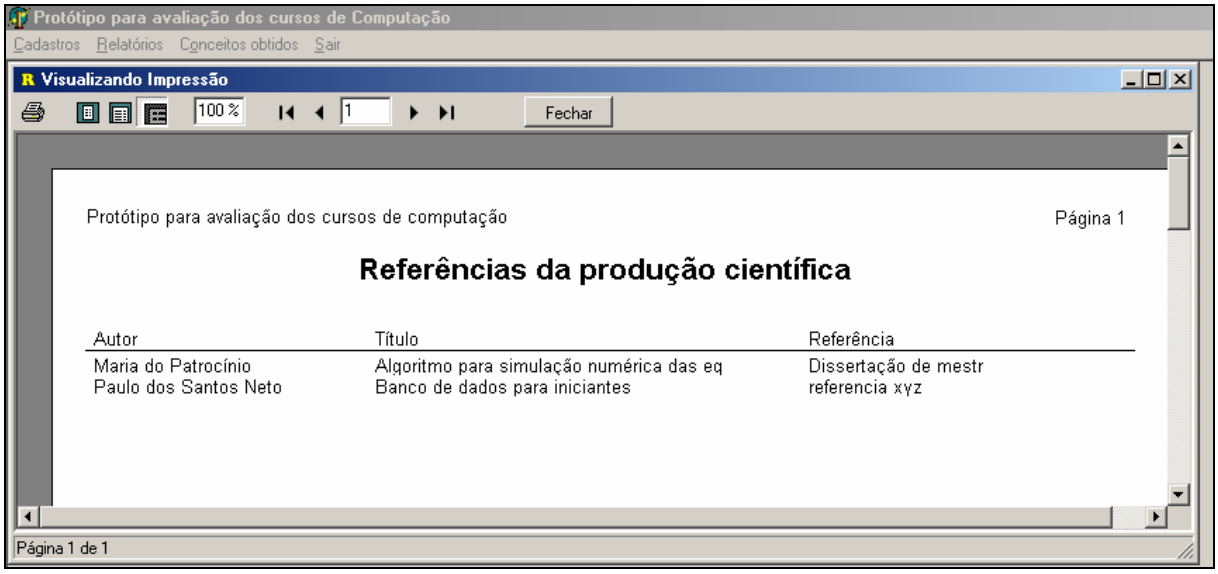
A figura 4.12 apresenta o relatório das disciplinas ministradas. Este deve imprimir somente as disciplinas ministradas nos últimos cinco anos.

FIGURA 4.12: relatório das disciplinas ministradas



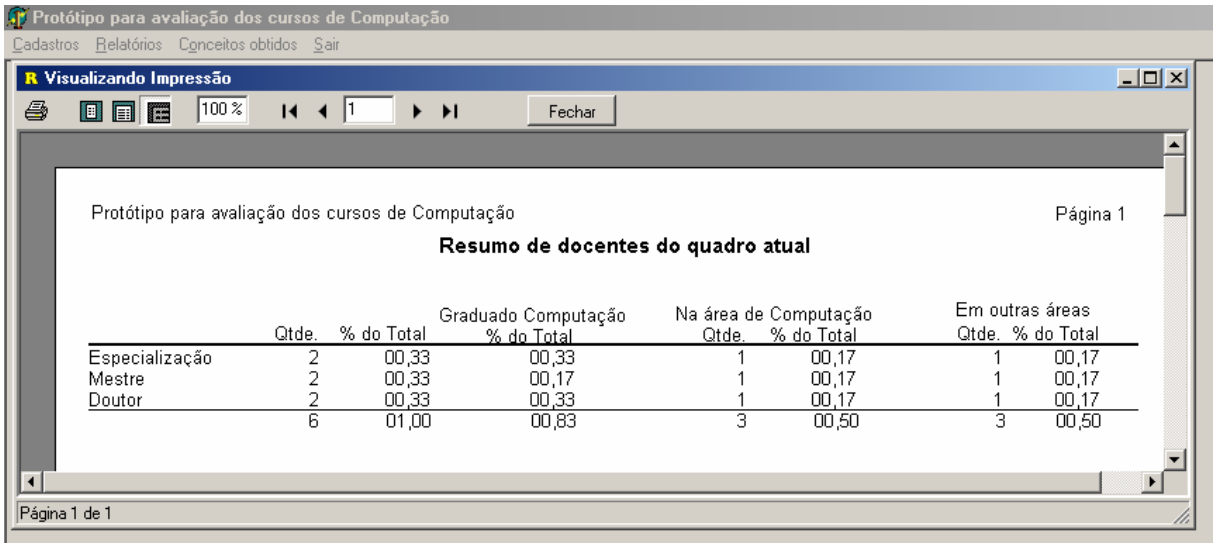
A figura 4.13 apresenta o relatório com a produção científica de todos os docentes.

FIGURA 4.13: Relatório da produção científica dos docentes



A figura 4.14 apresenta o relatório com o resumo do quadro atual dos docentes.

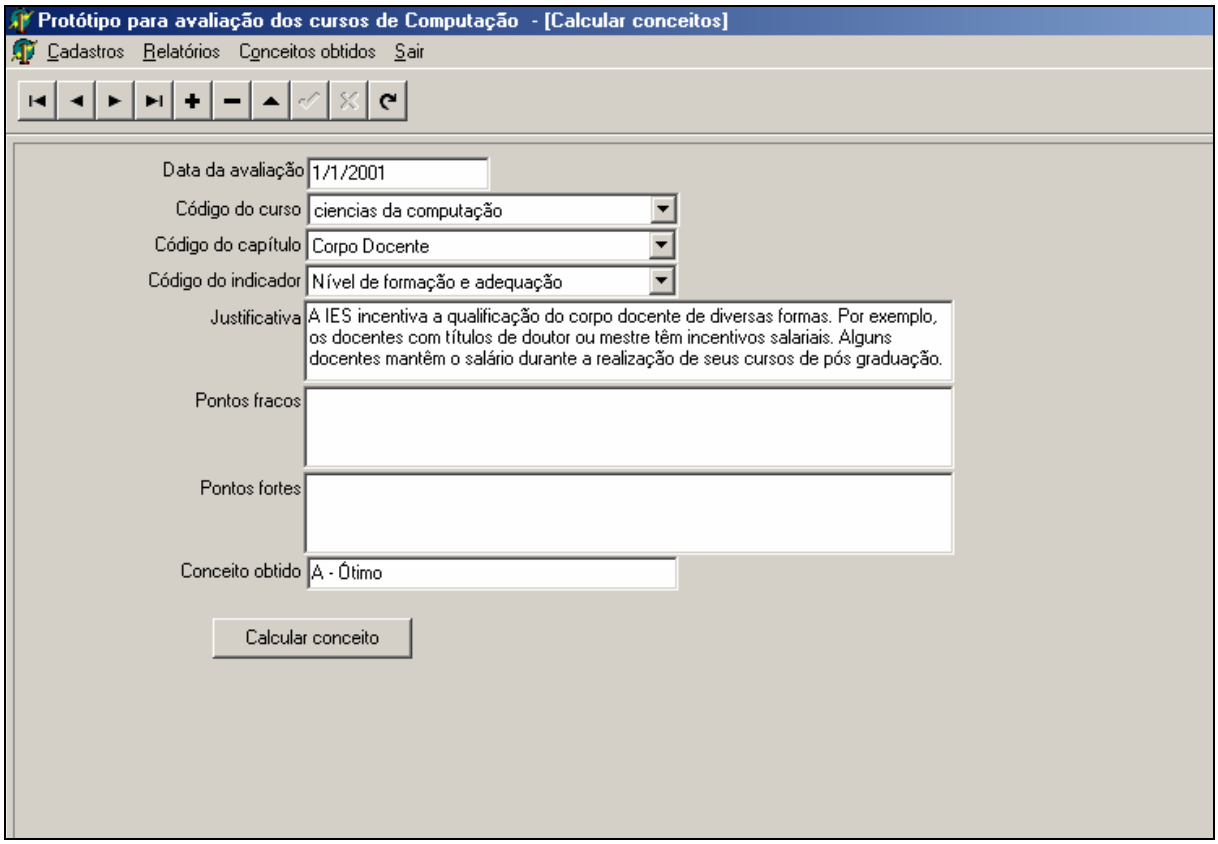
FIGURA 4.14: Relatório com resumo do quadro atual dos docentes



	Qtde.	% do Total	Graduado Computação % do Total	Na área de Computação Qtde.	% do Total	Em outras áreas Qtde.	% do Total
Especialização	2	00,33	00,33	1	00,17	1	00,17
Mestre	2	00,33	00,17	1	00,17	1	00,17
Doutor	2	00,33	00,33	1	00,17	1	00,17
	6	01,00	00,83	3	00,50	3	00,50

Após todos os dados preenchidos pode ser feita a avaliação do indicador ‘nível de formação e adequação do corpo docente’. Isto é calculado baseado em padrões definidos pelo MEC para cada indicador. O conceito é gravado e mostrado na tela da figura 4.15, podendo o conceito ser alterado devido a entrevistas feitas com os docentes e após o cadastramento dos pontos fortes, pontos fracos e justificativas do conceito obtido.

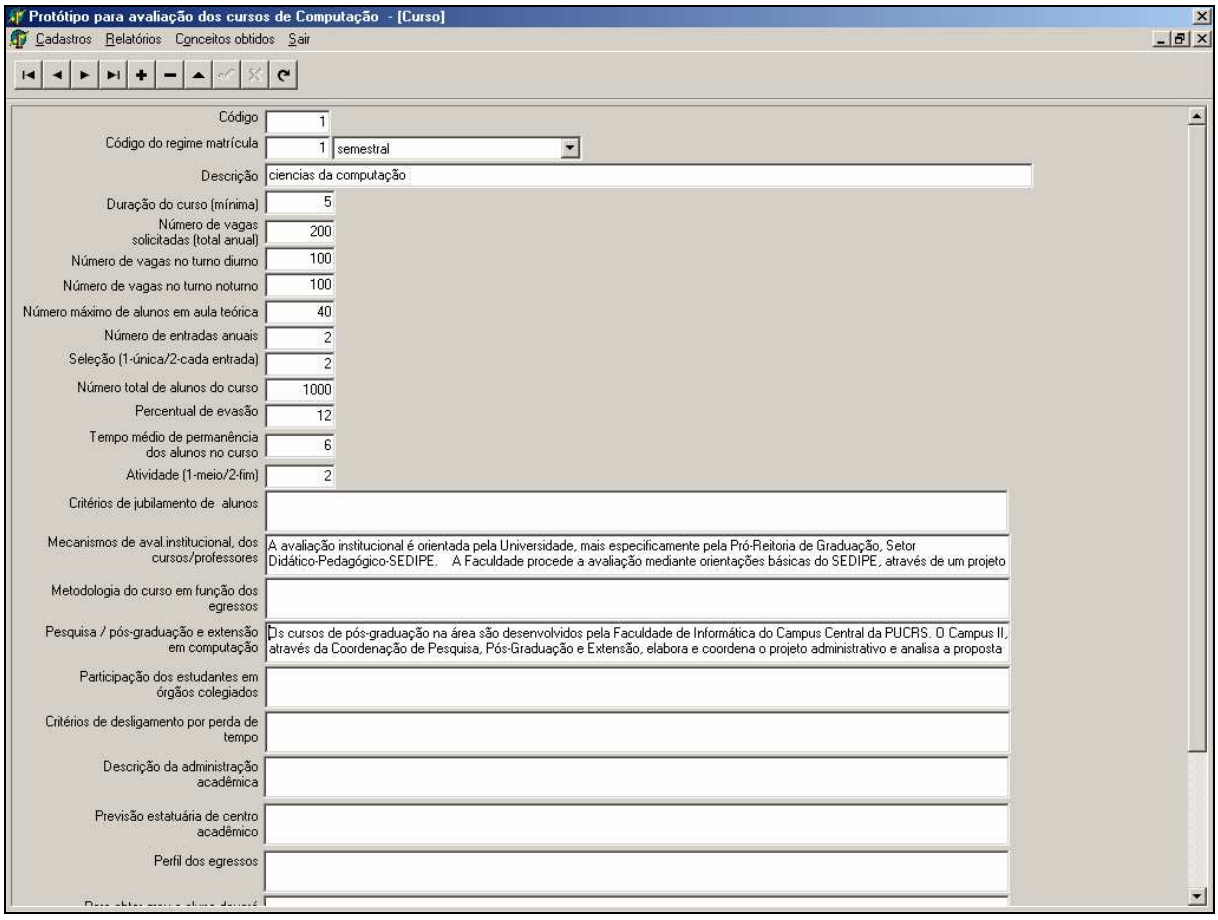
FIGURA 4.15: Tela da avaliação do conceito obtido



O exemplo descrito na figura 4.16 é apenas o indicador de um capítulo. O protótipo avalia quatro capítulos divididos em indicadores.

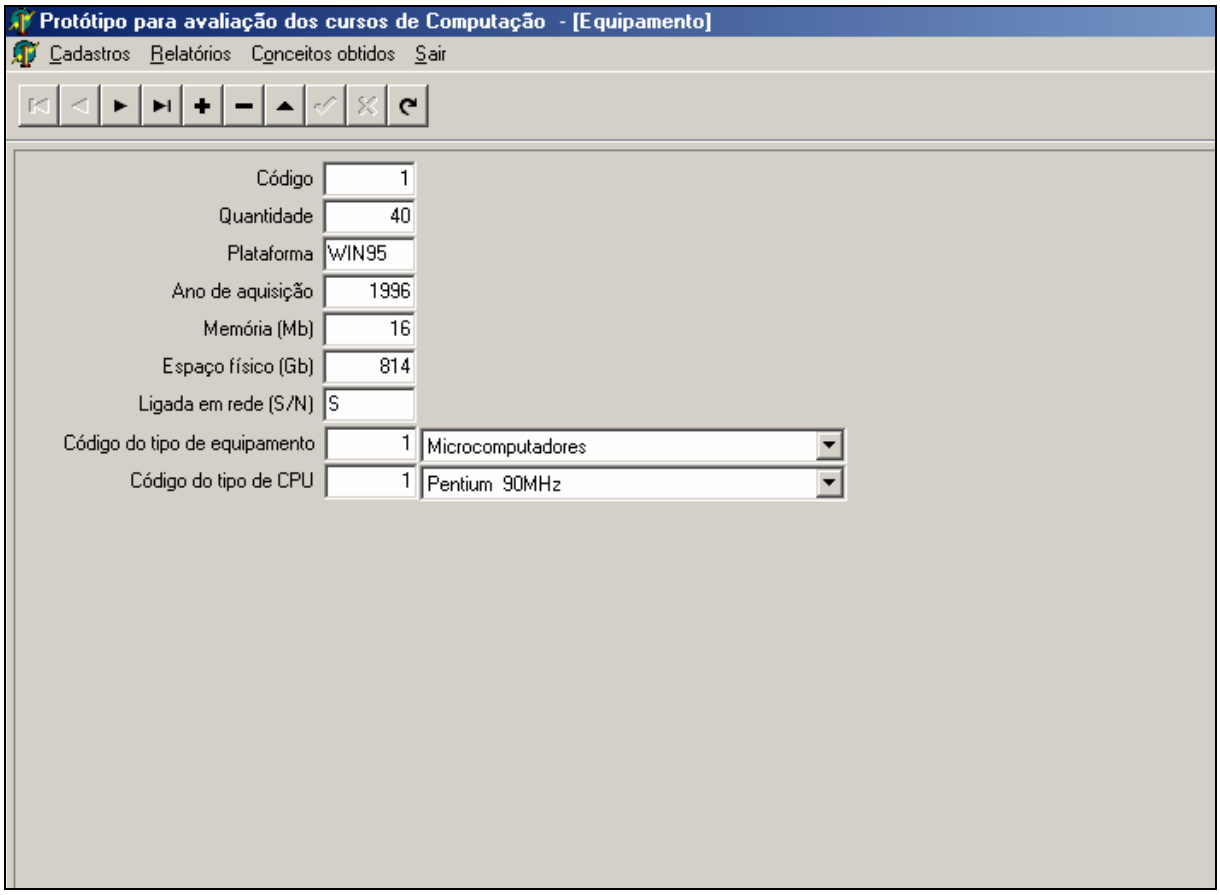
O segundo capítulo avalia o Plano Pedagógico, com diversos cadastros de informações referentes ao curso. Por exemplo, por curso são cadastrados critérios de desligamento de alunos (ver cadastro do curso na figura 6.8).

FIGURA 4.16 : Tela de cadastro de curso



O terceiro capítulo avalia a Infra-estrutura da Instituição, com informações sobre a Biblioteca e a infra-estrutura interna da Instituição. A figura 4.17 apresenta o cadastro de equipamentos e a figura 4.19 apresenta o cadastro do pessoal técnico de apoio para o curso.

FIGURA 4.17: Tela de cadastro de equipamentos



O quarto capítulo avalia o desempenho do curso. Neste capítulo são feitos um levantamento de Exame Nacional de cursos (ver figura 4.18 do cadastro) e um levantamento de desempenho do curso em relação a número de inscritos, tempo médio de permanência dos alunos no curso, entre outras informações.

FIGURA 4.18: Tela de cadastro do Exame Nacional de cursos

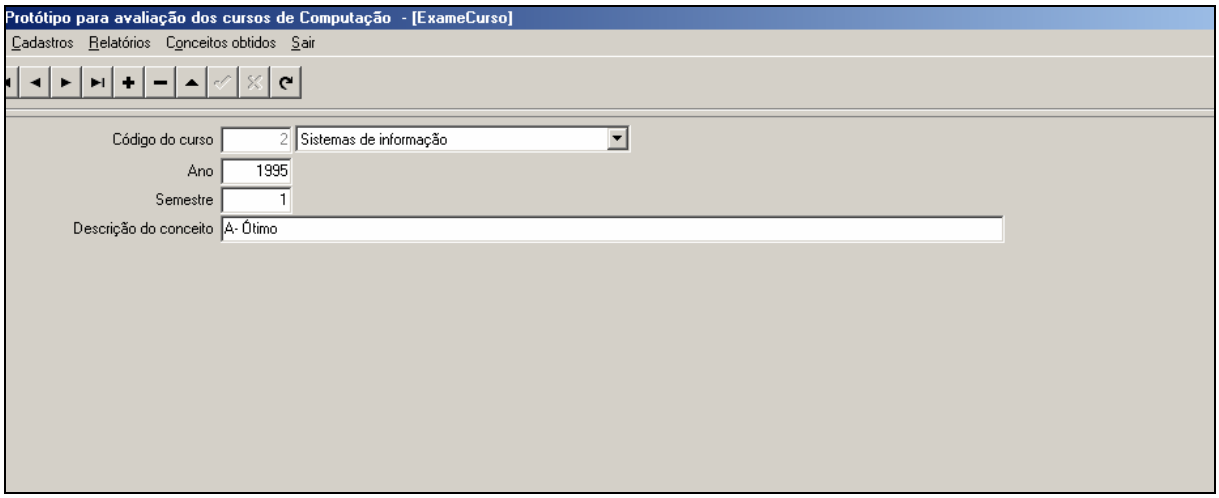
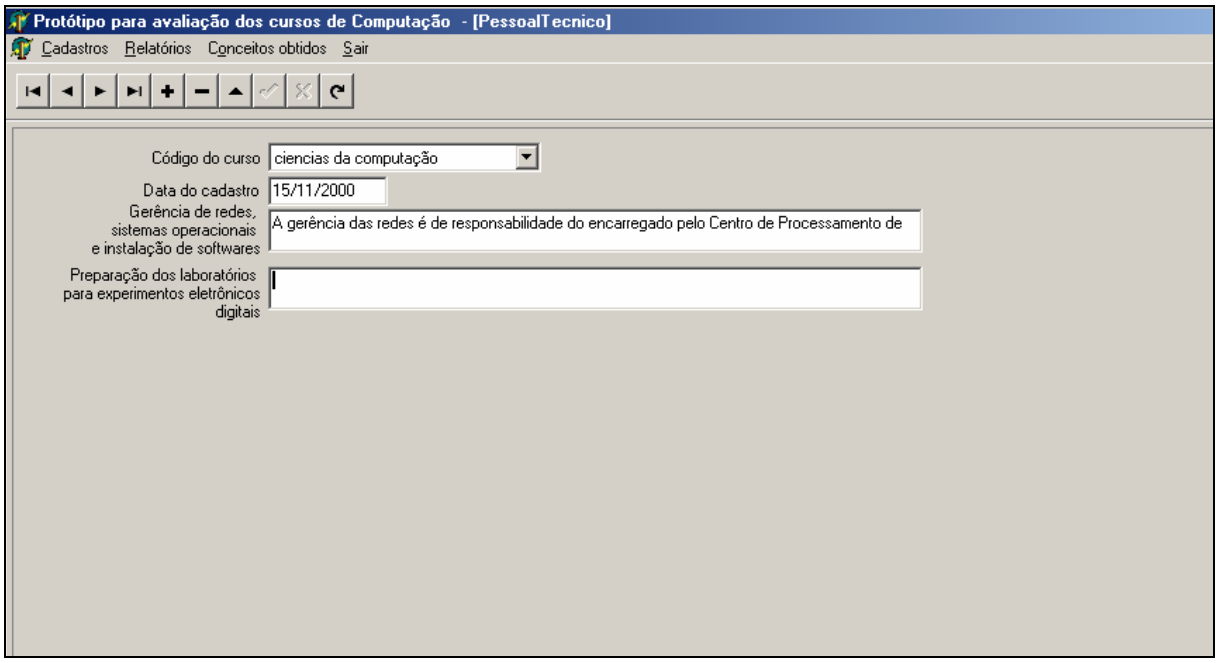


FIGURA 4.19: Tela de cadastros de pessoal técnico



A seguir será mostrado, a prototipação da ferramenta CASE. A figura 4.20 apresenta a tela de cadastro de regime de trabalho que a própria ferramenta gerou e a figura 4.21 apresenta o código gerado para esta tela.

FIGURA 4.20: Tela de cadastro de regime de trabalho

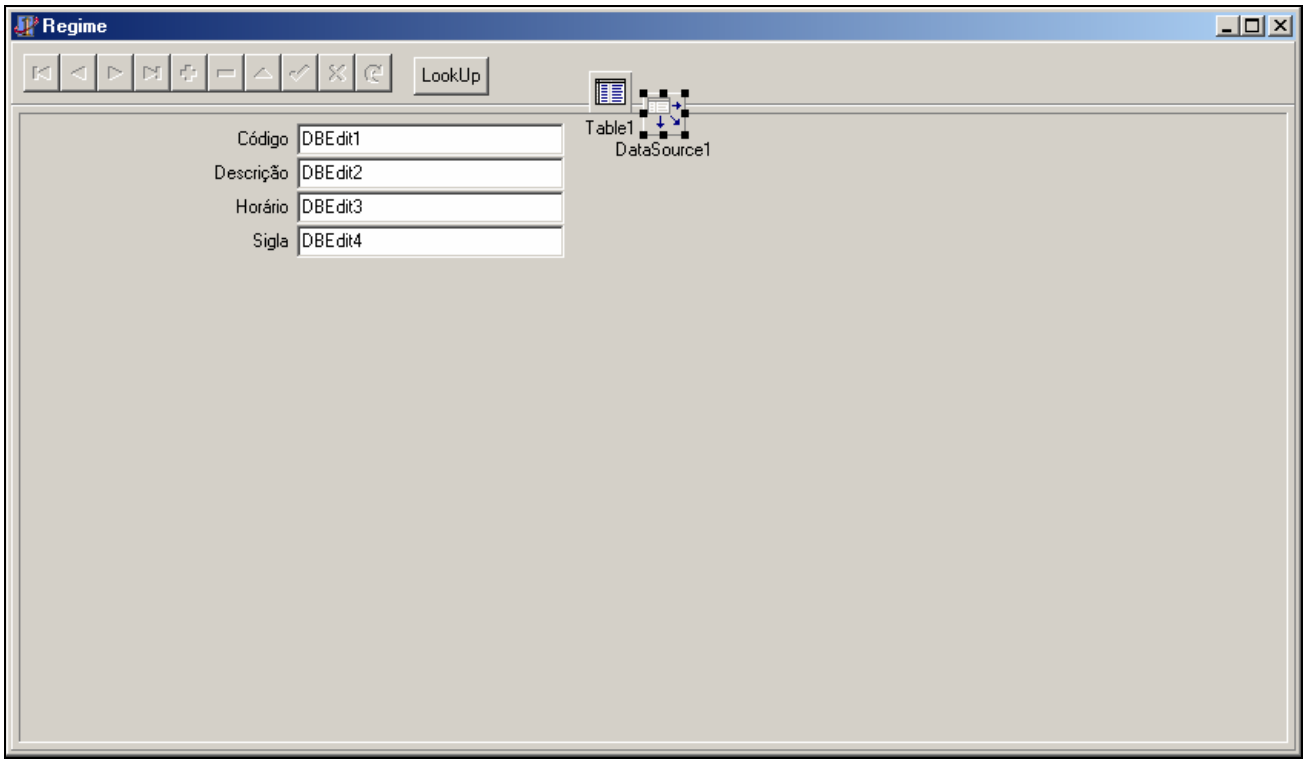


FIGURA 4.21: Parte de código gerado na prototipação pela ferramenta CASE

```
unit Unit29;
interface
uses SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  StdCtrls, Forms, DBCtrls, DB, DBTables, Mask, ExtCtrls;
type
  TForm29 = class(TForm)
    ScrollBox: TScrollBox;
    DBNavigator: TDBNavigator;
    Panel1: TPanel;
    DataSource1: TDataSource;
    Panel2: TPanel;
    Table1: TTable;
    LookUp: TButton;
    Label1: TLabel;
    DBEdit1: TDBEdit;
    Label2: TLabel;
    DBEdit2: TDBEdit;
    Label3: TLabel;
    DBEdit3: TDBEdit;
    Label4: TLabel;
    DBEdit4: TDBEdit;
    procedure FormCreate(Sender: TObject);
    procedure LookUpClick(Sender: TObject);
  private
```

```
    { private declarations }
  public
    { public declarations }
  end;

var
  Form29: TForm29;

implementation

{$R *.DFM}

uses Lookup;

procedure TForm29.FormCreate(Sender: TObject);
begin
  Table1.Open;
end;

procedure TForm29.LookUpClick(Sender: TObject);
begin
  FLookup:=TFLookup.create(self);
  FLookup.table1.active:=false;
  FLookup.table1.TableName:=table1.tablename;
  FLookup.table1.active:=true;
  FLookup.table1.gotocurrent(table1);
  FLookup.caption:='LookUp from '+caption;
  FLookup.ShowModal;
  FLookup.Free;
end;

end.
```

Todas as demais telas de cadastro são geradas da mesma forma e para cada tabela da base de dados a ferramenta cria um tela de cadastro para a tabela.

O protótipo implementado e gerado parcialmente pela ferramenta CASE *System Architect* não foi mais detalhado em função de que o foco do trabalho é a avaliação da ferramenta CASE. O protótipo construído auxiliou nesta avaliação, visto que vários recursos da ferramenta puderam ser testados no desenvolvimento de uma aplicação real.

5 AVALIAÇÃO DA FERRAMENTA CASE

Neste tópico será feita a avaliação da ferramenta CASE mostrando o que foi verificado com o uso da ferramenta CASE. O significado das características e das sub-características da norma utilizada nesta avaliação já foi abordado no capítulo 3. Para facilitar a leitura optou-se por incluir as observações e comentários sobre cada sub-característica de forma objetiva. Para auxiliar na avaliação das características foi utilizado um questionário que está no anexo 2. Ao final de cada característica avaliada é apresentado um gráfico de pizza para uma melhor visualização do resultado da avaliação. O gráfico mostra o percentual obtido na avaliação de cada característica. Para calcular este percentual foi utilizado o questionário do anexo 2 e os comentários sobre as sub-características.

5.1 PROCESSO DE CICLO DE VIDA

5.1.1 DESENVOLVIMENTO DE DIAGRAMA

A ferramenta permite a importação de modelos entre enciclopédias.

A parte de desenho (edição, alteração, arrastar, mover) diagramas, a ferramenta atende completamente. Por exemplo, é possível selecionar vários objetos de um diagrama e mover para outro lugar no diagrama sem perda de informações.

No diagrama de classes quando se faz a exclusão de um objeto, o seu relacionamento permanece, o correto seria ao se retirar um objeto os seus relacionamentos automaticamente serem excluídos. E no modelo entidade-relacionamento ao se retirar um objeto ao qual a chave primária do mesmo era uma chave estrangeira, a chave estrangeira permanece.

Através do modelo entidade-relacionamento se gera o diagrama de classes da UML, onde todas as tabelas ficam fora do posicionamento após a *Forward and Reverse Code Engineer* e a ferramenta coloca em cada relacionamento uma seta, o que não é padrão da UML.

A ferramenta suporta todos os diagramas da UML. O diagrama de seqüência e o diagrama de colaboração são consistentes. Inclusive, pode-se optar por efetuar as alterações apenas num deles, que serão refletidas no outro pela ferramenta.

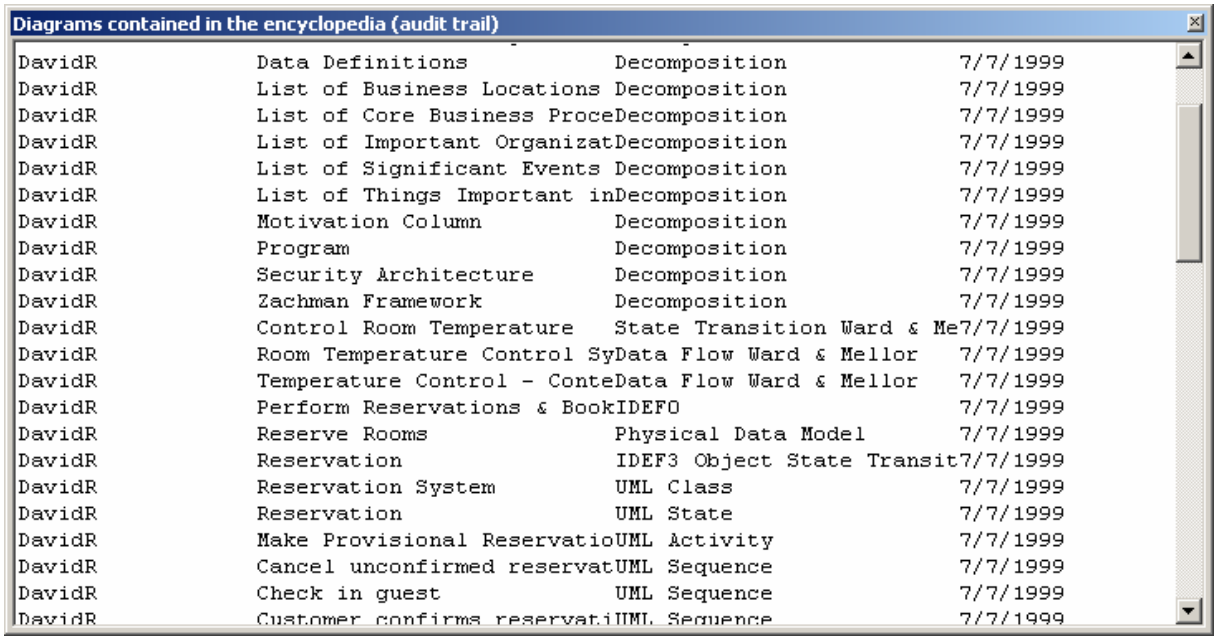
São independentes o modelo entidade-relacionamento e o modelo físico. Isto permite que o analista faça o modelo lógico com suas características de normalização e não redundância, mantendo sua coerência lógica.

É possível criar mais de um modelo físico a partir de um único modelo lógico, e isto é importante para quem trabalha com mais de um SGBD.

5.1.2 ANÁLISE DE DIAGRAMA

A ferramenta CASE avaliada possui um gerador de relatórios onde é possível fazer uma análise minunciosa dos diagramas. Por exemplo, o relatório da figura 5.1 mostra uma relação com todos os tipos de diagramas do projeto, indicando quem fez a última alteração (primeira coluna), a data (última coluna), o nome do diagrama (segunda coluna) e o seu tipo (terceira coluna).

FIGURA 5.1: Exemplo de um relatório do *System Architect*

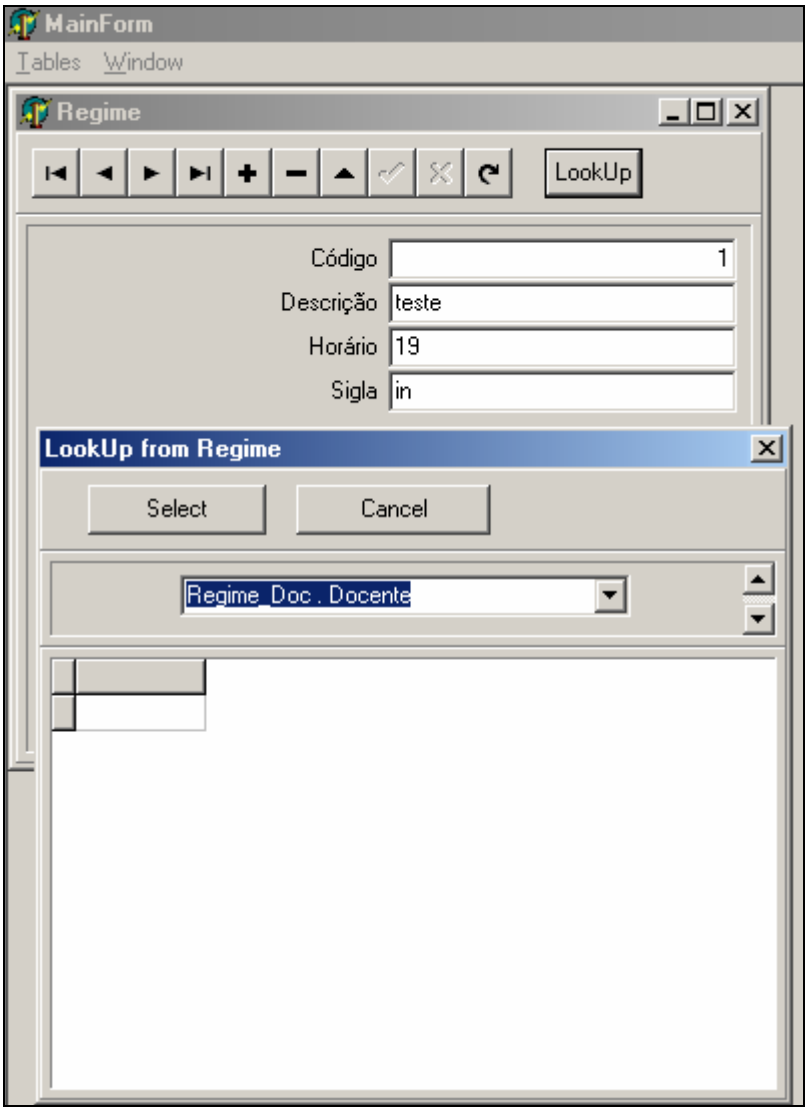


5.1.3 PROTOTIPAÇÃO

Na parte de telas de cadastros, é gerado um botão chamado ‘lookup’, o que permite visualizar informações de outras tabelas. Isto é gerado automaticamente. Por exemplo, no

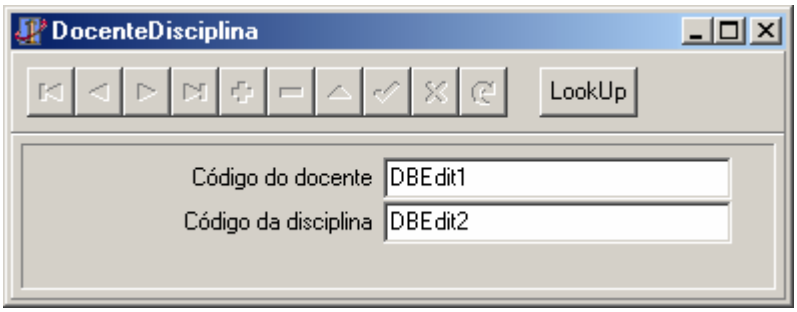
cadastro de regime de trabalho deste protótipo, o botão ‘lookup’ abre outra tela com todos os docentes pertencentes ao regime de trabalho selecionado. Conforme figura 5.2.

FIGURA 5.2: Exemplo de cadastro do protótipo



Geração de uma tela de cadastro para cada tabela, inclusive para tabelas de relacionamentos. Por exemplo, as tabelas Docente, Disciplina e DocenteDisc que é a tabela que informa que um docente pode lecionar mais de uma disciplina também cria uma tela de cadastro, o que é desnecessário. Ver figura 5.3.

FIGURA 5.3: Exemplo do protótipo



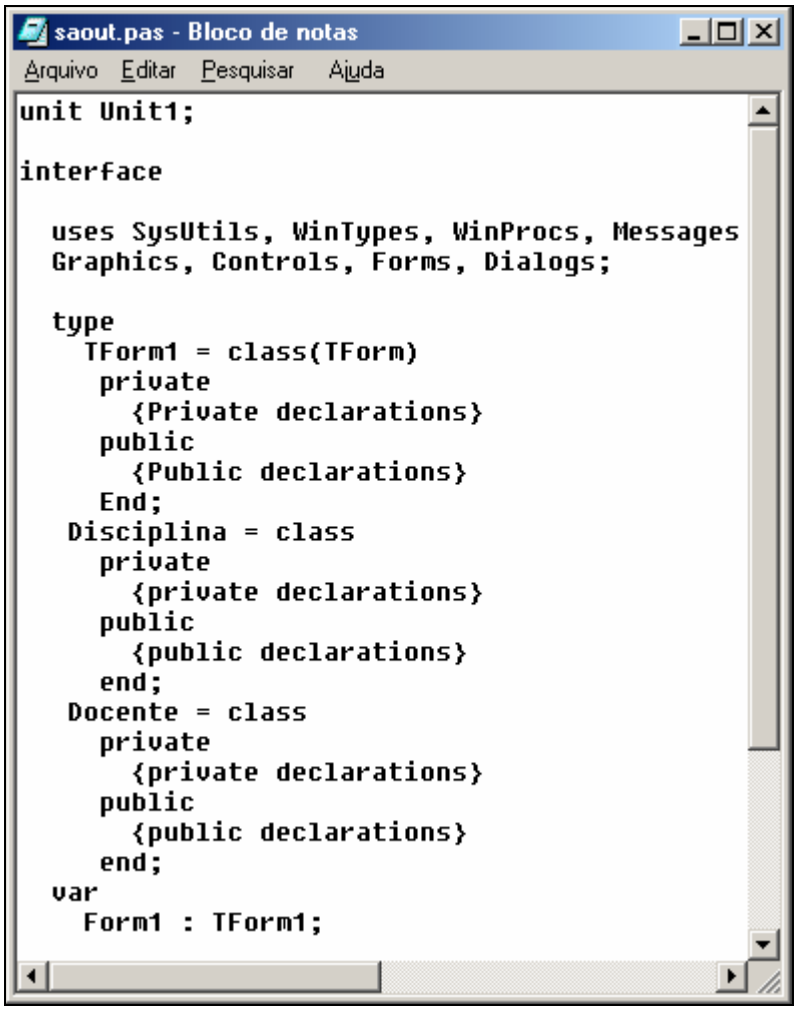
As Units são geradas com o nome: ‘unit+número seqüencial’, o que dificulta muito quando da consulta de um determinado formulário. Sugere-se incluir ao nome da unit o nome da tabela, devido ao fato das units serem de telas de cadastros.

5.1.4 GERAÇÃO DE CÓDIGO

Existe a possibilidade de desenvolver scripts de geração de código para outras linguagens, utilizando o VBA no *System Architect*. Sua geração é efetuada com poucos passos, em relação a outras ferramentas.

As linguagens suportadas para a geração de código são Smalltalk, C++, Java, VB, Delphi, Corba. Na figura 5.4 é mostrado um exemplo de código gerado para o Delphi.

FIGURA 5.4: Exemplo da geração de código no *System Architect*



5.1.5 GERAÇÃO DE ESQUEMAS DE BANCO DE DADOS

O *script* é gravado num arquivo texto, o que permite a sua alteração após ter sido gerado (ver anexo 1, um exemplo de script gerado pela ferramenta CASE).

A geração script pode ser feita de várias formas, através de um arquivo DDL ou *script* .SQL ou através de ODBC.

Para a geração do script é necessário criar o modelo físico da base de dados, e para criar o modelo físico, é necessário criar o modelo entidade-relacionamento. Ou seja, para este

processo o System Architect recorre a técnicas estruturadas, mesmo sendo a modelagem orientada a objetos.

No script da base de dados não ocorre a criação do banco de dados, que é o primeiro passo antes da criação das tabelas, mesmo selecionada a opção ‘Create database’ nas opções da geração do script.

5.1.6 GERAÇÃO DE TELAS

O gerador de telas não possui ligação com a geração do código fonte, pois se o usuário aprovou a tela de apresentação feita na ferramenta, a mesma não é aproveitada na geração do código fonte, uma maneira de aproveitá-la é substituindo após a geração do código fonte pela tela criada na ferramenta.

A ferramenta não possui componentes de acesso ao banco de dados para a geração da tela. Se o sistema utiliza banco de dados, as telas de apresentação necessitarão de alterações de seus componentes para isso.

Os componentes de acesso à banco de dados, possuem uma opção de ‘tamanho máximo’. Este quando da geração das telas, vem com um valor fixo. Por exemplo, no script um campo descrição possui tamanho de 40 posições, na tela a opção ‘tamanho máximo’ está com o tamanho de 20 posições, muito inferior ao tamanho definido no script.

5.1.7 GERAÇÃO DE RELATÓRIOS

A ferramenta não automatiza o desenvolvimento de relatórios a serem produzidos por sistemas em desenvolvimento.

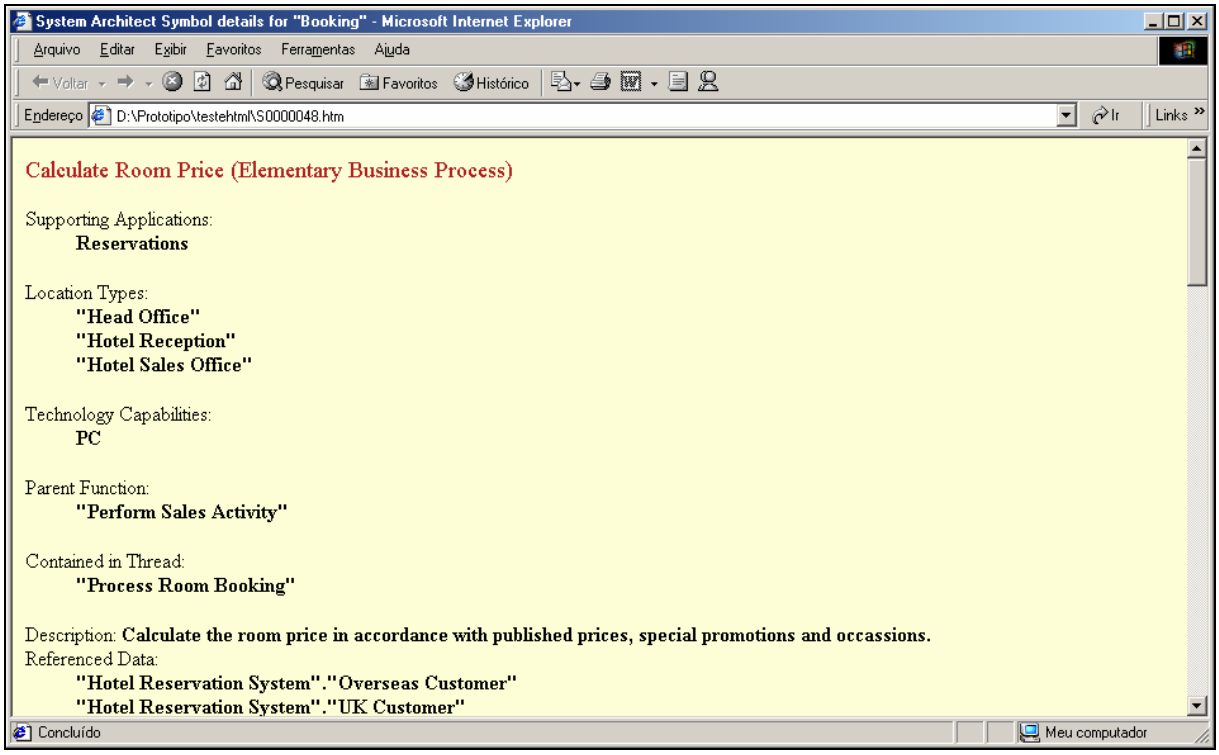
5.1.8 ENGENHARIA REVERSA DE DADOS

Suporta a engenharia para várias linguagens, entre elas, Oracle, Interbase, Sybase, Informix, SQL Server e por drive ODBC. Este recurso não foi testado, porém obteve-se informação do manual e *help* da ferramenta.

5.1.9 SUPORTE PARA HIPERTEXTO

A ferramenta possui gerador de relatórios em formato html e no Microsoft Word**.** Em formato html, há a opção de extrair dados de todos os tipos de diagramas, conforme figura 5.5.

FIGURA 5.5: Exemplo de geração relatório em formato HTML



5.1.10 EXTRAÇÃO AUTOMÁTICA DE DADOS E GERAÇÃO DE DOCUMENTAÇÃO

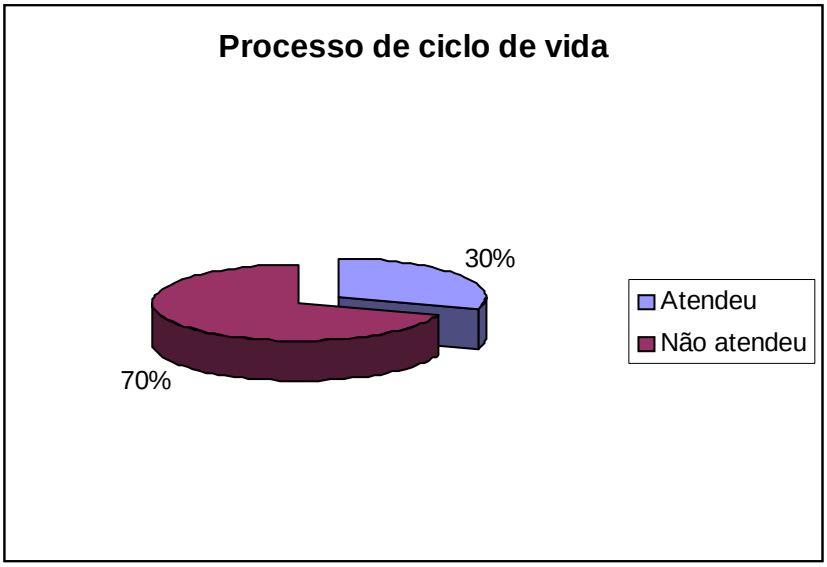
A ferramenta possui um gerador de relatórios próprio, oferecendo facilidades na elaboração de relatórios customizados. A partir do repositório, é possível extrair relatórios em função das informações contidas nele, como relatórios baseados em símbolos, nas definições, relacionamentos, regras de checagem, etc.

A ferramenta possui ainda relatórios padrões em função da técnica escolhida. Por exemplo, na UML tem-se um conjunto de relatórios próprios.

5.1.11 GRÁFICO DO PROCESSO DE CICLO DE VIDA DO SOFTWARE

O gráfico da figura 5.6 mostra o resultado da avaliação desta característica.

FIGURA 5.6: Gráfico do resultado da avaliação



5.2 USO DA FERRAMENTA CASE

5.2.1 CARACTERÍSTICAS DE HARDWARE REQUERIDO PELA FERRAMENTA

O espaço físico mínimo para a instalação da ferramenta é de 50Mb, um processador PC Pentium 133, memória mínima de 32MB (64Mb recomendado), conforme indicação no site do fabricante. Os ambientes suportados são as plataformas Windows 9x / NT / 2000.

5.2.2 AMBIENTE DE SOFTWARE REQUERIDO PELA FERRAMENTA

A ferramenta suporta apenas as plataformas Windows 9x/NT/2000. Em redes, através de versões multi-usuários, numa plataforma de comunicação à qual o Windows se conecte, permitindo o uso em grupos de trabalho através do compartilhamento das áreas de trabalho.

5.2.3 REPOSITÓRIO DE SOFTWARE (BASE DE INFORMAÇÃO)

O repositório pode ser compartilhado entre vários analistas. A ferramenta permite a criação de informações (tabelas, atributos e relacionamentos) através do repositório, porém não é possível gerar diagramas baseados nestas informações criadas através do repositório.

5.2.4 COMPATIBILIDADE COM ELEMENTOS DO AMBIENTE

A ferramenta permite a importação e exportação de dados no formato CSV (separação por vírgulas), no formato texto e através da engenharia reversa.

5.2.5 INTEGRAÇÃO DE DADOS

Suporta a exportação / importação de dados usando formatos universais (CSV), pode ser por repositório, através do desenvolvimento em XMI utilizando a UML.

5.2.6 AMBIENTE DE HARDWARE E SOFTWARE DOS PRODUTOS DA FERRAMENTA

A ferramenta suporta ser executada em plataformas RISC/SISC.

A ferramenta possibilita a comunicação com alguns dos softwares mais conhecidos no mercado, como Powersoft Power Builder v.4.0 ou posterior, Delphi 2.0 ou posterior, Microsoft Visual Basic v. 4.0 ou posterior, Microsoft Word v. 6.0 ou posterior, Intersolv PVCS 5.2 ou superior.

5.2.7 TAMANHO DE APLICAÇÃO SUPORTADO

Existe um limite para tamanho (mínimo e máximo) de diagrama da UML. Este tamanho porém é suficiente para se modelar diagramas complexos. É possível manter vários diagramas abertos ao mesmo tempo na ferramenta.

5.2.8 LINGUAGENS SUPORTADAS

A ferramenta suporta um número considerável de linguagens: Java, Java script, C++, Delphi, Visual Basic, Smalltalk, Corba IDL, Power Builder, HTML. Para o protótipo utilizou-se a linguagem Delphi.

5.2.9 BANCOS DE DADOS SUPORTADOS

A ferramenta suporta um número considerável de banco de dados: Access, AS/400, SQLBASE, DB2, Informix, Interbase, Oracle, Paradox, Progress, SQL Server, Sybase, Teradata, XDB. Considera-se como um ponto forte, porque não são poucos os banco de dados suportados pela ferramenta. Para o protótipo utilizou-se o banco de dados *Interbase*.

5.2.10 SUPORTE METODOLÓGICO

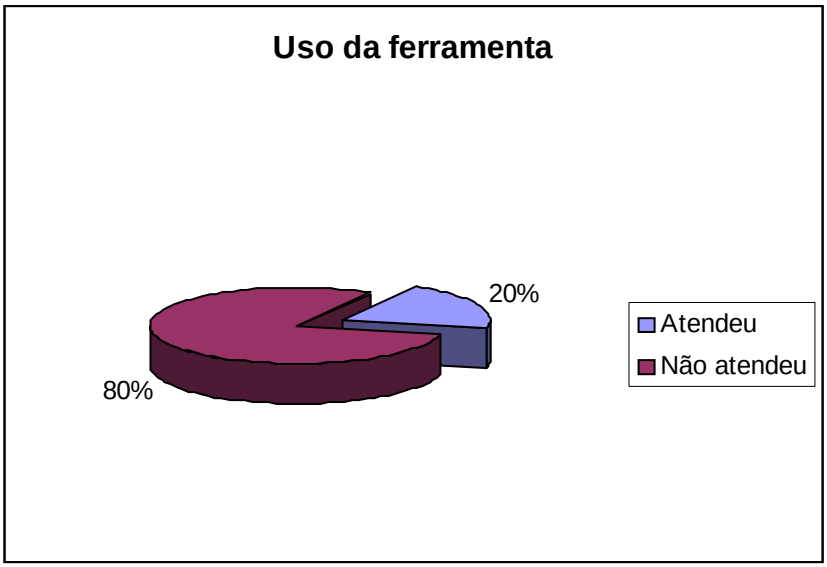
O suporte metodológico da ferramenta avaliada é vasto. As metodologias se dividem em estruturadas e orientadas a objetos. Alguns métodos estruturados: Gane & Sarson, Engenharia da Informação, Yourdon/De Marco, Ward & Mellor. Alguns métodos orientados a objetos: Booch 94, Coad Yourdon, OMT – Rumbaugh, Shlaer/Mellor, UML (1.3).

A ferramenta não obriga o desenvolvedor a utilizar somente uma metodologia no seu projeto, permitindo num mesmo projeto que seja possível utilizar técnicas de diferentes abordagens, em função da parametrização efetuada quando se cria um projeto. Neste trabalho foram usados o diagrama entidade-relacionamento e os diagramas da UML

5.2.11 GRÁFICO DO USO DA FERRAMENTA CASE

O gráfico da figura 5.7 mostra o resultado da avaliação desta característica.

FIGURA 5.7: Gráfico do resultado da avaliação



5.3 CARACTERÍSTICAS GERAIS DE QUALIDADE

5.3.1 SEGURANÇA

O System Architect faz a segurança através do *Bindery Utility*, que auxilia na proteção de acesso não autorizado. Ela possibilita até seis tipos de níveis de restrições guardados em arquivos com extensão .DLL. Por exemplo, no arquivo SABIND1.DLL está definido o nível um de segurança, onde o usuário é sempre um supervisor de todos os *drivers* e o usuário tem acesso a todos os *paths*. No arquivo SABIND6.DLL está definido o nível seis da segurança, onde cada usuário possui acesso a determinados *paths* definidos pelo *Bindery Utility*. O nível *default* é o dois, onde o usuário nunca é supervisor e tem acesso a todos os *paths*.

5.3.2 CONFORMIDADE TÉCNICA

A ferramenta possui conformidade técnica a algumas notações, por exemplo, os métodos Booch 94, Coad Yourdon, OMT – Rumbaugh, Shlaer/Mellor, UML (1.3).

5.3.3 INTEGRIDADE DE DADOS

A ferramenta CASE não suporta a utilização de informações de outras ferramentas CASE.

5.3.4 TOLERÂNCIA À FALHAS

Nesta versão demonstração, ocorreu uma falha, um erro numa DLL e logo em seguida da mensagem de erro, a ferramenta é fechada automaticamente sem salvar. Não houve nenhum tipo de prevenção a este erro e nenhum tipo de correção, como por exemplo, o salvamento antes de fechar a ferramenta.

5.3.5 RECUPERABILIDADE

Falta a opção para poder voltar a alteração efetuada. A ferramenta não oferece este recurso para que o usuário desfaça os processos efetuados.

5.3.6 INTELIGIBILIDADE

A ferramenta possui inteligibilidade, pois os padrões definidos na UML para o desenho de seus diagramas foram todos especificados adequadamente.

5.3.7 OPERACIONALIDADE

A ferramenta possui vários tipos de arquivos de LOG. Durante a execução de qualquer processo, a ferramenta é clara e concisa de forma que o usuário saiba realmente o que está acontecendo, pois a cada passo de um processo executado, uma tela com a execução é mostrada, ou seja, o usuário não fica sem saber o que está acontecendo. Como por exemplo, durante o processo de geração do modelo físico, uma tela com a criação do mesmo é mostrada na tela.

Não é possível utilizar a ferramenta enquanto ela estiver gerando, por exemplo, o código fonte, pois antes da geração a ferramenta fecha automaticamente todos as janelas abertas.

5.3.8 MANIPULAÇÃO DE ERROS

Ao encontrar qualquer tipo de erro durante a execução de processos a ferramenta para a execução mostrando o erro, que é facilmente interpretável.

5.3.9 FACILIDADE DE APRENDIZAGEM

O *Help On-Line* (inglês) e tutoriais (inglês) da ferramenta, explicam passo a passo como criar diagramas, o que facilita o aprendizado. Durante este trabalho foram obtidas apostilas de cursos oferecidos na região, que também auxiliaram na aprendizagem da ferramenta.

5.3.10 QUALIDADE DA DOCUMENTAÇÃO

O *Help On-Line* (inglês), tutoriais (inglês) e manuais técnicos relacionados aos diagramas da UML, possuem bons exemplos e explicações que foram buscados de um projeto de Reservas de Hotel que a própria ferramenta disponibilizou como exemplo prático.

5.3.11 FACILIDADE DE INSTALAÇÃO

A ferramenta não possui dificuldade para instalação, a qual é feita por CD. Gastou-se cerca de 30 minutos para a instalação.

5.3.12 TEMPO DE RESPOSTA

Para avaliar o tempo de resposta, depende muito do hardware usado para a avaliação. Numa máquina PC Pentium 133 Mhz (que é a recomendação mínima), a execução de todos os processos não é lenta. O que mais demora em relação a todos os outros é a geração das telas.

5.3.13 REQUISITOS DE ARMAZENAMENTO DE DADOS

O espaço necessário para o armazenamento da ferramenta é mínimo. A ferramenta instalada ocupa em torno de 50MB e o script de uma base de dados ocupa em torno de 30kb, este tamanho de script é o do protótipo. O que ocupa um pouco mais de espaço são os diagramas, por serem de visualização gráfica.

5.3.14 CAPACIDADE DE MEMÓRIA ACEITÁVEL

O espaço físico mínimo para a instalação é de 50Mb, um processador PC Pentium 133 Mhz, memória mínima de 32MB (64Mb recomendado), conforme consulta ao site do fornecedor.

5.3.15 PORTABILIDADE PARA DIFERENTES PLATAFORMAS DE HARDWARE

Esta ferramenta pode ser executada em plataformas RISC/SISC entre outras.

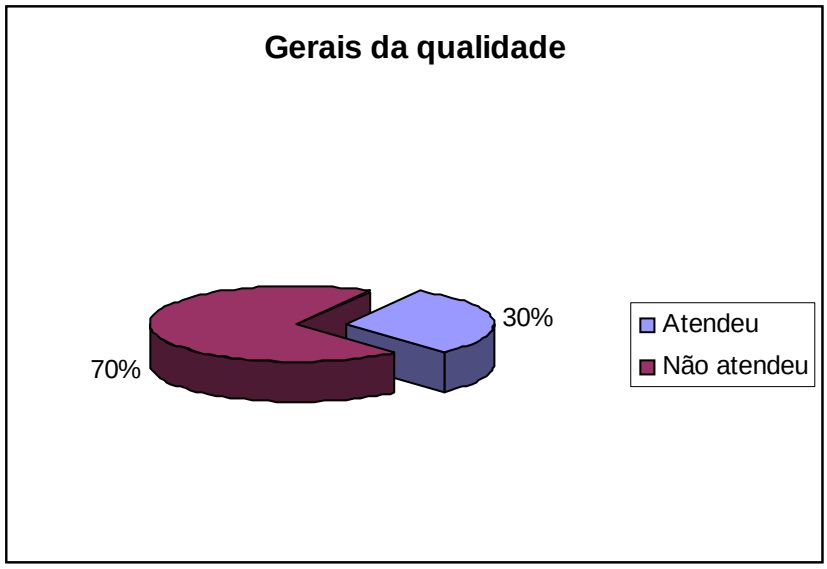
5.3.16 COMPATIBILIDADE ENTRE DIFERENTES SISTEMAS OPERACIONAIS

Esta ferramenta pode ser executada nos sistemas operacionais Windows 9x/NT/2000.

5.3.17 GRÁFICO DAS CARACTERÍSTICAS GERAIS DA QUALIDADE

O gráfico da figura 5.8 mostra o resultado da avaliação desta característica.

FIGURA 5.8: Gráfico do resultado da avaliação



5.4 CARACTERÍSTICAS GERAIS NÃO RELACIONADAS À QUALIDADE

5.4.1 PERFIL DO FORNECEDOR

Na região de Blumenau existe um fornecedor para a ferramenta, que é a empresa *Choose Technologies* que existe desde 1991 e possue várias filiais espalhadas pelo Brasil (São Paulo, Brasíla, Belo Horizonte, Blumenau, Porto Alegre, Recife, Vitória, Curitiba, Salvador e Rio de Janeiro).

Além do *System Architect* o fornecedor é revendedor exclusivo de algumas ferramentas de apoio ao desenvolvimento, como, *StarTeam*, ILOG.

Faz treinamento metodológico e gerencial, consultoria em *Capability Maturity Model for Software* (CMM), Metodologias de Desenvolvimento de Sistemas (MDS) e Modelagem de Processos de Negócio (MPN).

5.4.2 PERFIL DO PRODUTO

Via Internet é possível instalar com facilidade uma cópia da ferramenta, uma versão demo para 30 dias.

A aquisição de uma cópia demonstrativa da ferramenta, é simples, envia-se um e-mail para um fornecedor próximo.

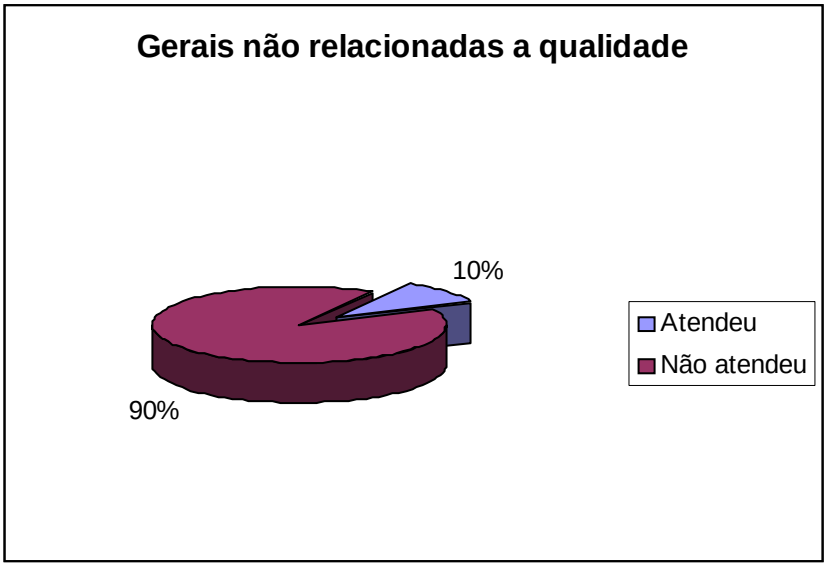
5.4.3 DISPONIBILIDADE DE TREINAMENTO

Não há um profissional para fazer treinamento na região de Blumenau, há mais profissionais disponíveis nas grandes capitais.

5.4.4 GRÁFICO DAS CARACTERÍSTICAS GERAIS NÃO RELACIONADAS A QUALIDADE

O gráfico da figura 5.9 mostra o resultado da avaliação desta característica.

FIGURA 5.9: Gráfico do resultado da avaliação



5.5 COMENTÁRIOS SOBRE O CASE

Analisando o desenvolvimento do protótipo com o CASE tem-se alguns comentários:

a) para se obter uma visão geral do protótipo, utilizou-se o diagrama de Caso de Uso;

b) para uma expansão e maior detalhamento dos principais processos do protótipo mostrados no diagrama de Caso de Uso, utilizou-se o diagrama de seqüência ao qual explica o funcionamento dos processos e principalmente mostra onde estão as informações necessárias. Porém, este diagrama apresentou problemas de desenho que não impediram seu uso;

c) para modelar a estrutura do protótipo, utilizou-se o diagrama de Classes. Este diagrama apresentou sérios problemas, já relatados no item da avaliação ‘desenvolvimento de diagrama’. Com estes problemas, perdeu-se muito tempo para consertá-los;

d) para especificar operações utilizou-se o diagrama de Atividade que são usados como fluxogramas para especificar um cálculo, por exemplo, os cálculos dos conceitos dos indicadores de qualidade. Este diagrama atendeu perfeitamente os objetivos de especificar os cálculos, porém, em relação ao desenho apresentou sérios problemas, já relatados no item da avaliação ‘desenvolvimento de diagramas’. Com estes problemas, cada vez que se consultam os diagramas é preciso reposicioná-los para poder visualizá-los, o que ocasiona perda de tempo.

e) para a geração do script da base de dados, foi preciso gerar o modelo físico que é baseado no modelo entidade-relacionamento. Ocorreu apenas um problema na geração do script do protótipo, não tinha a criação do banco;

f) foi gerado o protótipo, baseado no modelo entidade-relacionamento, optando-se pela linguagem Object Pascal. Ocorreram alguns problemas, relatados no item ‘prototipação’, todos tiveram que ser corrigidos para continuar o desenvolvimento do protótipo;

g) o protótipo possui vários relatórios, nesta etapa a ferramenta não ajudou. Pois a ferramenta CASE *System Architect* não possui nenhum tipo de auxílio para o desenvolvimento de relatórios do protótipo.

6 CONCLUSÕES

Através deste trabalho foi possível aprimorar o conhecimento sobre a orientação a objetos e a UML. Outro conhecimento adquirido ao longo do trabalho foi no uso da ferramenta CASE *System Architect*.

O protótipo para auto-avaliação dos cursos de Computação atingiu os objetivos de avaliar um curso conforme definições do MEC. O objetivo principal deste trabalho foi atendido, que foi avaliar a qualidade da ferramenta CASE *System Architect* utilizando características previstas na norma ISO/IEC 14102.

A avaliação foi feita analisando-se pontos fortes e pontos fracos da ferramenta, que indica por exemplo, que na prototipação para o Object Pascal a ferramenta avaliada não traz um resultado satisfatório para o desenvolvimento rápido de sistemas. Um outro ponto fraco da ferramenta, acontece na modelagem UML, onde o diagrama de Classes é gerado pela ferramenta através do Diagrama entidade-relacionamento. O resultado da geração necessita de muitas correções feitas manualmente. Um ponto fraco desta ferramenta, citado no capítulo dois sobre vantagens das ferramentas CASE por Silva e Videira (2001), é a geração automática de código. O código gerado é pouco aproveitado na sua forma original para o desenvolvimento de um sistema.

Na fase de análise e especificação do protótipo e também na fase da documentação, a ferramenta CASE atende a seus objetivos. As telas do protótipo foram em partes geradas pela ferramenta e incorporadas no código fonte, com diversas alterações. O tempo gasto no desenvolvimento sem o auxílio da CASE, foi maior na parte de relatórios do protótipo e gastou-se muito tempo para corrigir problemas que a ferramenta gerou na modelagem dos dados. Parte destes problemas até foram ocasionados por falhas na concepção do modelo.

Um ponto forte é em relação aos diagramas da UML, onde só é necessário se fazer o desenho do diagrama de seqüência, pois o diagrama de colaboração é gerado automaticamente pela ferramenta baseado no diagrama de seqüência. Um outro ponto positivo da ferramenta é em relação a segurança, que é feita através do *Bindery Utility*, que auxilia na proteção de acesso não autorizado.

6.1 EXTENSÕES

Sugere-se como extensão para trabalhos futuros nesta área, a avaliação da ferramenta CASE System Architect de forma mais aprofundada utilizando-se outras técnicas de análise.

Outra sugestão é avaliar outras ferramentas CASE, utilizando de forma completa a norma ISO/IEC 14102.

ANEXO 1 – SCRIPT GERADO PELA FERRAMENTA CASE

```
CREATE DATABASE "d:\prototipo\banco\banco5.gdb" USER "SYSDBA" PASSWORD "masterkey";
CREATE TABLE AlunosCurso(
        CDCURSO                 INTEGER NOT NULL,
        nrAno                   INTEGER NOT NULL,
        nrSemestre              INTEGER NOT NULL,
        nrTotVagas              INTEGER,
        nrFormados              INTEGER,
        nrInscritos             INTEGER);
ALTER TABLE AlunosCurso ADD
   CONSTRAINT AlunosCurso_PK PRIMARY KEY (CDCURSO,nrAno,nrSemestre);
CREATE TABLE AvaliacaoCapitulo(
        cdCurso                 INTEGER NOT NULL,
        dtAvaliacao             DATE NOT NULL,
        CDCAPITULO              INTEGER,
        CDINDICADOR             INTEGER,
        CDCONCEITO              INTEGER,
        dsJustificativa         BLOB,
        dsPontosFracos          BLOB,
        dsPontosFortes          BLOB);
ALTER TABLE AvaliacaoCapitulo ADD
   CONSTRAINT AvaliacaoGrupo_PK PRIMARY KEY (dtAvaliacao,cdCurso);
CREATE TABLE Biblioteca(
        dtCadastro              DATE NOT NULL,
        dsPeriodicos            BLOB,
        dsPoliticaAtualiz       BLOB,
        dsHrAcesso              CHARACTER(40),
        dsFormaAcesso           CHARACTER(40),
        dsFormaEmprestimo       CHARACTER(40),
        dsReservas              BLOB NOT NULL,
        dsCatalogAcervo         BLOB,
        dsDispAcervo            BLOB,
        dsReprografia           BLOB,
        dsEspacoFisico          BLOB,
        cdInstituicao           INTEGER NOT NULL);
ALTER TABLE Biblioteca ADD
   CONSTRAINT Biblioteca_PK PRIMARY KEY (dtCadastro);
CREATE TABLE Capitulo(
        cdCapitulo              INTEGER NOT NULL,
        dsCapitulo              CHARACTER(30) NOT NULL,
        dsJustificativa         BLOB);
ALTER TABLE Capitulo ADD
   CONSTRAINT Capitulo_PK PRIMARY KEY (cdCapitulo);
CREATE TABLE CargaHoraria(
        CDDISCIPLINA            INTEGER NOT NULL,
        CDDOCENTE               INTEGER NOT NULL,
        nrAno                   INTEGER NOT NULL,
        nrSemestre              INTEGER NOT NULL,
        cdTurma                 CHARACTER(2) NOT NULL,
        nrMatricula             INTEGER,
        nrHrSemana              INTEGER NOT NULL,
        nrVagasOfer             INTEGER);
ALTER TABLE CargaHoraria ADD
   CONSTRAINT CargaHoraria_PK PRIMARY KEY
(CDDISCIPLINA,CDDOCENTE,nrAno,nrSemestre,cdTurma);
```

```
CREATE TABLE ComissaoAvaliacao(
        CDCURSO                 INTEGER NOT NULL,
        nrPortaria              INTEGER,
        dsEspecialista          CHARACTER(80),
        dsApoioTecnico          CHARACTER(80),
        dtVisitaIni             DATE,
        dtVisitaFim             DATE,
        dtPortaria              DATE,
        dtPublicacaoDOU         DATE);
ALTER TABLE ComissaoAvaliacao ADD
   CONSTRAINT ComissaoAvaliacao_PK PRIMARY KEY (CDCURSO);
CREATE TABLE Conceito(
        cdconceito              INTEGER NOT NULL,
        dsConceito              CHARACTER(30));
ALTER TABLE Conceito ADD
   CONSTRAINT Conceito_PK PRIMARY KEY (cdconceito);
CREATE TABLE Coordenador(
        dtIniCoordenacao        DATE NOT NULL,
        dtFimCoordenacao        DATE,
        dsExperiencia           BLOB,
        CDDOCENTE               INTEGER NOT NULL);
ALTER TABLE Coordenador ADD
   CONSTRAINT Coordenador_PK PRIMARY KEY (CDDOCENTE);
CREATE TABLE Curso(
        cdCurso                 INTEGER NOT NULL,
        CDREGIMEMAT             INTEGER,
        dsCurso                 CHARACTER(30),
        dsPoliticaAperf         BLOB,
        dsPerfilEgressos        BLOB,
        dsMetodoEgressos        BLOB,
        dsObterGrau             BLOB,
        dsCritJubila            BLOB,
        dsCritReprov            BLOB,
        dsCritPerdaTmp          BLOB,
        dsPesqExtensao          BLOB,
        dsAdmAcad               BLOB,
        dsParticEstudo          BLOB,
        dsPrevisaoEst           BLOB,
        nrMinimoDuracao         INTEGER,
        nrVagasSolicAno         INTEGER,
        nrVagasDiurno           INTEGER,
        nrVagasNoturno          INTEGER,
        nrAlunosTeorico         INTEGER,
        nrEntAnuais             INTEGER,
        dsSelecaoEnt            INTEGER,
        nrTotAlunos             INTEGER,
        nrPercEvasao            INTEGER,
        dsAcompEgresso          BLOB,
        dsMECAvalIES            BLOB,
        nrTmpMedioPerman        INTEGER,
        dsAtividade             NUMERIC(1));
ALTER TABLE Curso ADD
   CONSTRAINT Curso_PK PRIMARY KEY (cdCurso);
CREATE TABLE CursoDocente(
        cdCurso                 INTEGER NOT NULL,
        cdDocente               INTEGER NOT NULL);
```

```
ALTER TABLE CursoDocente ADD
   CONSTRAINT CursoDocente_PK PRIMARY KEY (cdCurso,cdDocente);
CREATE TABLE Disciplina(
        cdDisciplina                 INTEGER NOT NULL,
        CDNIVEL1                     INTEGER,
        CDNIVEL2                     INTEGER,
        CDNIVEL3                     INTEGER,
        CDNIVEL4                     INTEGER,
        nmDisciplina                 CHARACTER(30),
        nrCreditos                   INTEGER,
        dsEmenta                     BLOB,
        nrCargaHoraria               INTEGER,
        nrSemestreCurr               INTEGER,
        cdcarater                    CHARACTER(1) NOT NULL,
        idExigeFormacaoComputacao    CHARACTER(1) NOT NULL);
ALTER TABLE Disciplina ADD
   CONSTRAINT Disciplina_PK PRIMARY KEY (cdDisciplina);
CREATE TABLE DisciplinaMinistrada(
        cdDocente                    INTEGER NOT NULL,
        cdDisciplina                 INTEGER NOT NULL,
        dtLecionada                  DATE,
        nmInstituicao                CHARACTER(40),
        dsNivel                      CHARACTER(30));
ALTER TABLE DisciplinaMinistrada ADD
   CONSTRAINT DisciplinaMinistrada_PK PRIMARY KEY (cdDocente,cdDisciplina);
CREATE TABLE Docente(
        cdDocente                    INTEGER NOT NULL,
        nmDocente                    CHARACTER(40) NOT NULL,
        CDREGIME                     INTEGER,
        nrCPF                        CHARACTER(15) NOT NULL,
        nrCGC                        CHARACTER(15) NOT NULL,
        dsOrgaoEmissor               CHARACTER(10) NOT NULL,
        dtEmissao                    DATE NOT NULL,
        nrFax                        CHARACTER(15),
        nrFone                       CHARACTER(15) NOT NULL,
        edResidencial                CHARACTER(25) NOT NULL,
        edEMail                      CHARACTER(25),
        dsAreaAtuacao                BLOB NOT NULL,
        dtAdmissao                   DATE NOT NULL,
        dtRescisao                   DATE,
        dsAtivDiscente               CHARACTER VARYING(500) NOT NULL,
        idTemEstabilidade            CHARACTER(1) NOT NULL,
        cdEnquadramento              CHARACTER(2) NOT NULL,
        IDGraduadoComputacao         CHARACTER(1) NOT NULL);
ALTER TABLE Docente ADD
   CONSTRAINT Docente_PK PRIMARY KEY (cdDocente);
CREATE TABLE DocenteDisciplina(
        cdDocente                    INTEGER NOT NULL,
        cdDisciplina                 INTEGER NOT NULL);
ALTER TABLE DocenteDisciplina ADD
   CONSTRAINT DocenteDisciplina_PK PRIMARY KEY (cdDocente,cdDisciplina);
CREATE TABLE DocentePublica(
        cdDocente                    INTEGER NOT NULL,
        cdPublicacao                 INTEGER NOT NULL);
ALTER TABLE DocentePublica ADD
   CONSTRAINT DocentePublica_PK PRIMARY KEY (cdDocente,cdPublicacao);
```

```
CREATE TABLE Equipamento(
        Cdcurso                 INTEGER NOT NULL,
        cdEquipamento           INTEGER NOT NULL,
        CDTIPOCPU               INTEGER NOT NULL,
        qtEquipamento           INTEGER,
        idPlataforma            CHARACTER(30),
        nrAnoAquisicao          INTEGER,
        qtMemoria               INTEGER,
        qtEspacofisico          INTEGER,
        idLigadaRede            CHARACTER(1),
        dsEquipamento           CHARRACTER(90),
        qtTipoEquip             INTEGER,
        hrDispMaquina           INTEGER);
ALTER TABLE Equipamento ADD
   CONSTRAINT Equipamento_PK PRIMARY KEY (CDCURSO,cdEquipamento,CDTIPOCPU);
CREATE TABLE ExameCurso(
        CDCURSO                 INTEGER NOT NULL,
        nrAno                   INTEGER NOT NULL,
        nrSemestre              INTEGER NOT NULL,
        dsConceito              CHARACTER(30));
ALTER TABLE ExameCurso ADD
   CONSTRAINT ExameCurso_PK PRIMARY KEY (CDCURSO,nrAno,nrSemestre);
CREATE TABLE Indicador(
        cdsIndicador            INTEGER NOT NULL,
        dsIndicador             CHARACTER(40));
ALTER TABLE Indicador ADD
   CONSTRAINT Indicador_PK PRIMARY KEY (cdsIndicador);
CREATE TABLE InfraEstrutura(
        cdCurso                 INTEGER NOT NULL,
        dtCadastro              DATE NOT NULL,
        dsEspacoLabCmput        BLOB,
        dsEspacoLabHard         BLOB,
        dsEspacoBiblioteca      BLOB,
        dsSalaCurso             BLOB,
        dsAreaLazer             BLOB,
        dsAreaCirculacao        BLOB,
        dsAreaSanitarios        BLOB,
        dsRecursosAudio         BLOB,
        dsAdequacaoInstal       BLOB,
        dsFacilidDefFisicos     BLOB,
        dsInfraEstruturaCoord   BLOB,
        dsSalaEnsinoEspecializado BLOB);
ALTER TABLE InfraEstrutura ADD
   CONSTRAINT InfraEstrutura_PK PRIMARY KEY (dtCadastro,cdCurso);
CREATE TABLE Instituicao(
        cdInstituicao           INTEGER NOT NULL,
        nmMantida               CHARACTER(80) NOT NULL,
        nmMantenedora           CHARACTER(80) NOT NULL,
        edInstituicao           CHARACTER(80),
        dsBairro                CHARACTER(25),
        nmMunicipio             CHARACTER(25) NOT NULL,
        nrFone                  CHARACTER(15) NOT NULL,
        nrFax                   CHARACTER(15),
        dsHisMantenedora        BLOB,
        dsHisMantida            BLOB);
ALTER TABLE Instituicao ADD
   CONSTRAINT Instituicao_PK PRIMARY KEY (cdInstituicao);
```

```
CREATE TABLE InstituicaoCurso(
        cdInstituicao                   INTEGER NOT NULL,
        cdCurso                         INTEGER NOT NULL);
ALTER TABLE InstituicaoCurso ADD
   CONSTRAINT InstituicaoCurso_PK PRIMARY KEY (cdInstituicao,cdCurso);
CREATE TABLE LabComputacao(
        dtCadastro                      DATE NOT NULL,
        dsPoliticaAcesso                CHARACTER(40) NOT NULL UNIQUE,
        dsHrFuncionamento               CHARACTER(40) NOT NULL UNIQUE,
        dsContratoManutencao            CHARACTER(40) NOT NULL UNIQUE,
        dsEstruturaLocalManutencao      CHARACTER(40) NOT NULL UNIQUE,
        dsAtualizacaoTecnologica        CHARACTER(40) NOT NULL UNIQUE,
        cdCurso                         INTEGER NOT NULL);
ALTER TABLE LabComputacao ADD
   CONSTRAINT LabComputacao_PK PRIMARY KEY (dtCadastro,cdCurso);
CREATE TABLE LabHardware(
        cdCurso                         INTEGER NOT NULL,
        dtCadastro                      DATE NOT NULL,
        dsQ1Basico                      BLOB,
        dsQ2AdequacaoEgresso            BLOB,
        nrHrDisponivelExclusivo         INTEGER,
        nrAlunosBancada                 INTEGER);
ALTER TABLE LabHardware ADD
   CONSTRAINT LabHardware_PK PRIMARY KEY (dtCadastro,cdCurso);
CREATE TABLE Livro(
        cdLivro                         INTEGER NOT NULL,
        tpLivro                         NUMERIC(1) NOT NULL,
        nmLivro                         CHARACTER(60),
        nrExemplares                    INTEGER);
ALTER TABLE Livro ADD
   CONSTRAINT DiscLivro_PK PRIMARY KEY (cdLivro);
CREATE TABLE LivrosDisciplina(
        CDDISCIPLINA                    INTEGER NOT NULL,
        CDLIVRO                         INTEGER NOT NULL);
ALTER TABLE LivrosDisciplina ADD
   CONSTRAINT LivrosDisciplina_PK PRIMARY KEY (CDDISCIPLINA,CDLIVRO);
CREATE TABLE OutrosCursos(
        dtCadastro                      DATE NOT NULL,
        nmCurso                         CHARACTER(30),
        nmInstituicao                   CHARACTER(40),
        nmMunicipio                     CHARACTER(25),
        qtMediaCandVaga                 INTEGER,
        cdCurso                         INTEGER NOT NULL);
ALTER TABLE OutrosCursos ADD
   CONSTRAINT OutrosCursos_PK PRIMARY KEY (dtCadastro,cdCurso);
CREATE TABLE Parametros_Rel(
        cdCurso                         INTEGER NOT NULL,
        nrProcesso                      INTEGER NOT NULL,
        dtEmissao                       DATE NOT NULL,
        tpObjetivo                      NUMERIC(1),
        tpPrimAval                      NUMERIC(1),
        dsAtoReconhec                   BLOB,
        dsAtoAutoriz                    BLOB,
        dsProcesso                      BLOB,
        dsNomeSugeridoCurso             CHARACTER(40) NOT NULL);
ALTER TABLE Parametros_Rel ADD
   CONSTRAINT Parametros_Rel_PK PRIMARY KEY (nrProcesso,dtEmissao,cdCurso);
```

```
CREATE TABLE PessoalTecnico(
        cdCurso                 INTEGER NOT NULL,
        dtCadastro              DATE NOT NULL,
        dsGerenciaE             BLOB,
        dsPreparaLab            BLOB);
ALTER TABLE PessoalTecnico ADD
   CONSTRAINT PessoalTecnico_PK PRIMARY KEY (dtCadastro,cdCurso);
CREATE TABLE PreRequisito(
        cdDisciplina            INTEGER NOT NULL,
        cdDisciplinaPre         INTEGER NOT NULL);
ALTER TABLE PreRequisito ADD
   CONSTRAINT PreRequisito_PK PRIMARY KEY (cdDisciplina,cdDisciplinaPre);
CREATE TABLE PublicacaoLivro(
        cdPublicacao            INTEGER NOT NULL,
        nmTitulo                CHARACTER(80),
        dsReferencia            CHARACTER(80));
ALTER TABLE PublicacaoLivro ADD
   CONSTRAINT PublicacaoLivro_PK PRIMARY KEY (cdPublicacao);
CREATE TABLE Regime(
        cdRegime                INTEGER NOT NULL,
        dsRegime                CHARACTER(20) NOT NULL,
        dsHorario               CHARACTER(20),
        dsSigla                 CHARACTER(2));
ALTER TABLE Regime ADD
   CONSTRAINT Regime_PK PRIMARY KEY (cdRegime);
CREATE TABLE RegimeMatricula(
        cdRegimeMat             INTEGER NOT NULL,
        dsRegimeMat             CHARACTER(25) NOT NULL);
ALTER TABLE RegimeMatricula ADD
   CONSTRAINT RegimeMatricula_PK PRIMARY KEY (cdRegimeMat);
CREATE TABLE Software(
        cdSoftware              INTEGER NOT NULL,
        nmSoftware              CHARACTER(30),
        nrLicenca               CHARACTER(10),
        nrCopias                INTEGER);
ALTER TABLE Software ADD
   CONSTRAINT Software_PK PRIMARY KEY (cdSoftware);
CREATE TABLE SoftwareDisc(
        cdDisciplina            INTEGER NOT NULL,
        cdSoftware              INTEGER NOT NULL);
ALTER TABLE SoftwareDisc ADD
   CONSTRAINT SoftwareDisc_PK PRIMARY KEY (cdDisciplina,cdSoftware);
CREATE TABLE Taxonomia(
        cdNivel1                INTEGER NOT NULL,
        cdNivel2                INTEGER NOT NULL,
        cdNivel3                INTEGER NOT NULL,
        cdNivel4                INTEGER NOT NULL,
        dsTaxonomia             CHARACTER(30));
ALTER TABLE Taxonomia ADD
   CONSTRAINT Taxonomia_PK PRIMARY KEY (cdNivel1,cdNivel2,cdNivel3,cdNivel4);
CREATE TABLE TipoCPU(
        cdTipoCPU               INTEGER NOT NULL,
        dsTipoCPU               CHARACTER(30));
ALTER TABLE TipoCPU ADD
   CONSTRAINT TipoCPU_PK PRIMARY KEY (cdTipoCPU);
CREATE TABLE Titulacao(
        cdTitulacao             INTEGER NOT NULL,
```

```
        dsTitulacao                 CHARACTER(30) NOT NULL);
ALTER TABLE Titulacao ADD
   CONSTRAINT Titulacao_2_PK PRIMARY KEY (cdTitulacao);
CREATE TABLE TitulacaoDocente(
        CDDOCENTE                    INTEGER NOT NULL,
        CDTITULACAO                  INTEGER NOT NULL,
        nrAnoConclusao               INTEGER NOT NULL,
        dsAreaTitulacao              CHARACTER(50) NOT NULL,
        nranoCAPES                   INTEGER NOT NULL,
        nrPortariaCAPES              INTEGER NOT NULL,
        nrApostilamento              INTEGER NOT NULL,
        nmInstConclusao              CHARACTER(30) NOT NULL,
        idAreaTitulacao              CHARACTER(1) NOT NULL);
ALTER TABLE TitulacaoDocente ADD
   CONSTRAINT Titulacao_PK PRIMARY KEY (CDDOCENTE,CDTITULACAO);
/* FK for reference to Curso through relation Curso_AluCur */
ALTER TABLE AlunosCurso ADD
   CONSTRAINT Curso_AluCur FOREIGN KEY (CDCURSO)
   REFERENCES Curso (cdCurso);
/* FK for reference to Capitulo through relation Capitulo_aval */
ALTER TABLE AvaliacaoCapitulo ADD
   CONSTRAINT Capitulo_aval FOREIGN KEY (CDCAPITULO)
   REFERENCES Capitulo (cdCapitulo);
/* FK for reference to Conceito through relation Conceito_Aval */
ALTER TABLE AvaliacaoCapitulo ADD
   CONSTRAINT Conceito_Aval FOREIGN KEY (CDCONCEITO)
   REFERENCES Conceito (cdconceito);
/* FK for reference to Indicador through relation Indicador_AvalCap */
ALTER TABLE AvaliacaoCapitulo ADD
   CONSTRAINT Indicador_AvalCap FOREIGN KEY (CDINDICADOR)
   REFERENCES Indicador (cdsIndicador);
/* FK for reference to Curso through relation FK_CursoAvaliacao */
ALTER TABLE AvaliacaoCapitulo ADD
   CONSTRAINT FK_CursoAvaliacao FOREIGN KEY (cdCurso)
   REFERENCES Curso (cdCurso);
/* FK for reference to Instituicao through relation FK_IESBiblioteca */
ALTER TABLE Biblioteca ADD
   CONSTRAINT FK_IESBiblioteca FOREIGN KEY (cdInstituicao)
   REFERENCES Instituicao (cdInstituicao);
/* FK for reference to Docente through relation Docente_CargaHor */
ALTER TABLE CargaHoraria ADD
   CONSTRAINT Docente_CargaHor FOREIGN KEY (CDDOCENTE)
   REFERENCES Docente (cdDocente);
/* FK for reference to Disciplina through relation Disc_CargaHor */
ALTER TABLE CargaHoraria ADD
   CONSTRAINT Disc_CargaHor FOREIGN KEY (CDDISCIPLINA)
   REFERENCES Disciplina (cdDisciplina);
/* FK for reference to Curso through relation Curso_ComissaoAval */
ALTER TABLE ComissaoAvaliacao ADD
   CONSTRAINT Curso_ComissaoAval FOREIGN KEY (CDCURSO)
   REFERENCES Curso (cdCurso);
/* FK for reference to Capitulo through relation Capitulo_Condicao */
ALTER TABLE CondicaoOferta ADD
   CONSTRAINT Capitulo_Condicao FOREIGN KEY (CAPITULO)
   REFERENCES Capitulo (cdCapitulo);
/* FK for reference to Conceito through relation Conceito_Condicao */
ALTER TABLE CondicaoOferta ADD
   CONSTRAINT Conceito_Condicao FOREIGN KEY (CONCEITO)
```

```
   REFERENCES Conceito (cdconceito);
/* FK for reference to Docente through relation Docente_Coordenador */
ALTER TABLE Coordenador ADD
   CONSTRAINT Docente_Coordenador FOREIGN KEY (CDDOCENTE)
   REFERENCES Docente (cdDocente);
/* FK for reference to RegimeMatricula through relation RegimeMat_IES */
ALTER TABLE Curso ADD
   CONSTRAINT RegimeMat_IES FOREIGN KEY (CDREGIMEMAT)
   REFERENCES RegimeMatricula (cdRegimeMat);
/* FK for reference to Curso through relation Curso_Docente */
ALTER TABLE CursoDocente ADD
   CONSTRAINT Curso_Docente FOREIGN KEY (cdCurso)
   REFERENCES Curso (cdCurso);
/* FK for reference to Docente through relation Docente_CurDoc */
ALTER TABLE CursoDocente ADD
   CONSTRAINT Docente_CurDoc FOREIGN KEY (cdDocente)
   REFERENCES Docente (cdDocente);
/* FK for reference to Taxonomia through relation FKTaxonomia */
ALTER TABLE Disciplina ADD
   CONSTRAINT FKTaxonomia FOREIGN KEY (CDNIVEL1,CDNIVEL2,CDNIVEL3,CDNIVEL4)
   REFERENCES Taxonomia (cdNivel1,cdNivel2,cdNivel3,cdNivel4);
/* FK for reference to Docente through relation Docente_DiscMinist */
ALTER TABLE DisciplinaMinistrada ADD
   CONSTRAINT Docente_DiscMinist FOREIGN KEY (cdDocente)
   REFERENCES Docente (cdDocente);
/* FK for reference to Disciplina through relation Disc_DiscMinist */
ALTER TABLE DisciplinaMinistrada ADD
   CONSTRAINT Disc_DiscMinist FOREIGN KEY (cdDisciplina)
   REFERENCES Disciplina (cdDisciplina);
/* FK for reference to Regime through relation FKRegime */
ALTER TABLE Docente ADD
   CONSTRAINT FKRegime FOREIGN KEY (CDREGIME)
   REFERENCES Regime (cdRegime);
/* FK for reference to Docente through relation Docente_DocDisc */
ALTER TABLE DocenteDisciplina ADD
   CONSTRAINT Docente_DocDisc FOREIGN KEY (cdDocente)
   REFERENCES Docente (cdDocente);
/* FK for reference to Disciplina through relation Disc_DocDisc */
ALTER TABLE DocenteDisciplina ADD
   CONSTRAINT Disc_DocDisc FOREIGN KEY (cdDisciplina)
   REFERENCES Disciplina (cdDisciplina);
/* FK for reference to PublicacaoLivro through relation PubLivro_DocPub */
ALTER TABLE DocentePublica ADD
   CONSTRAINT PubLivro_DocPub FOREIGN KEY (cdPublicacao)
   REFERENCES PublicacaoLivro (cdPublicacao);
/* FK for reference to Docente through relation Docente_Doc_pub */
ALTER TABLE DocentePublica ADD
   CONSTRAINT Docente_Doc_pub FOREIGN KEY (cdDocente)
   REFERENCES Docente (cdDocente);
/* FK for reference to TipoEquipamento through relation TpEquip_Equip */
ALTER TABLE Equipamento ADD
   CONSTRAINT TpEquip_Equip FOREIGN KEY (CDTIPOEQUIP)
   REFERENCES TipoEquipamento (cdTipoEquip);
/* FK for reference to TipoCPU through relation TpCPU_Equip */
ALTER TABLE Equipamento ADD
   CONSTRAINT TpCPU_Equip FOREIGN KEY (CDTIPOCPU)
   REFERENCES TipoCPU (cdTipoCPU);
/* FK for reference to Curso through relation Curso_ExameCur */
```

```
ALTER TABLE ExameCurso ADD
   CONSTRAINT Curso_ExameCur FOREIGN KEY (CDCURSO)
   REFERENCES Curso (cdCurso);
/* FK for reference to Curso through relation CursoInfraEstrutura */
ALTER TABLE InfraEstrutura ADD
   CONSTRAINT CursoInfraEstrutura FOREIGN KEY (cdCurso)
   REFERENCES Curso (cdCurso);
/* FK for reference to Instituicao through relation Instituicao_InsCur */
ALTER TABLE InstituicaoCurso ADD
   CONSTRAINT Instituicao_InsCur FOREIGN KEY (cdInstituicao)
   REFERENCES Instituicao (cdInstituicao);
/* FK for reference to Curso through relation Curso_InsCur */
ALTER TABLE InstituicaoCurso ADD
   CONSTRAINT Curso_InsCur FOREIGN KEY (cdCurso)
   REFERENCES Curso (cdCurso);
/* FK for reference to Curso through relation Curso_LabComputacao */
ALTER TABLE LabComputacao ADD
   CONSTRAINT Curso_LabComputacao FOREIGN KEY (cdCurso)
   REFERENCES Curso (cdCurso);
/* FK for reference to Curso through relation Curso_LabHardware */
ALTER TABLE LabHardware ADD
   CONSTRAINT Curso_LabHardware FOREIGN KEY (cdCurso)
   REFERENCES Curso (cdCurso);
/* FK for reference to Disciplina through relation Disc_LivrosDisc */
ALTER TABLE LivrosDisciplina ADD
   CONSTRAINT Disc_LivrosDisc FOREIGN KEY (CDDISCIPLINA)
   REFERENCES Disciplina (cdDisciplina);
/* FK for reference to Livro through relation Livro_LivrosDisc */
ALTER TABLE LivrosDisciplina ADD
   CONSTRAINT Livro_LivrosDisc FOREIGN KEY (CDLIVRO)
   REFERENCES Livro (cdLivro);
/* FK for reference to Curso through relation Curso_OutroCursos */
ALTER TABLE OutrosCursos ADD
   CONSTRAINT Curso_OutroCursos FOREIGN KEY (cdCurso)
   REFERENCES Curso (cdCurso);
/* FK for reference to Curso through relation CursoParametrosRel */
ALTER TABLE Parametros_Rel ADD
   CONSTRAINT CursoParametrosRel FOREIGN KEY (cdCurso)
   REFERENCES Curso (cdCurso);
/* FK for reference to Curso through relation Curso_PessoalTecnico */
ALTER TABLE PessoalTecnico ADD
   CONSTRAINT Curso_PessoalTecnico FOREIGN KEY (cdCurso)
   REFERENCES Curso (cdCurso);
/* FK for reference to Disciplina through relation FK_Disciplina_Principal */
ALTER TABLE PreRequisito ADD
   CONSTRAINT FK_Disciplina_Principal FOREIGN KEY (cdDisciplina)
   REFERENCES Disciplina (cdDisciplina);
/* FK for reference to Disciplina through relation FKPreRequisito */
ALTER TABLE PreRequisito ADD
   CONSTRAINT FKPreRequisito FOREIGN KEY (cdDisciplinaPre)
   REFERENCES Disciplina (cdDisciplina);
/* FK for reference to Disciplina through relation Disc_SoftDisc */
ALTER TABLE SoftwareDisc ADD
   CONSTRAINT Disc_SoftDisc FOREIGN KEY (cdDisciplina)
   REFERENCES Disciplina (cdDisciplina);
/* FK for reference to Software through relation Soft_SoftDisc */
ALTER TABLE SoftwareDisc ADD
   CONSTRAINT Soft_SoftDisc FOREIGN KEY (cdSoftware)
```

```
   REFERENCES Software (cdSoftware);
/* FK for reference to Docente through unnamed relation */
ALTER TABLE TitulacaoDocente ADD
   FOREIGN KEY (CDDOCENTE)
   REFERENCES Docente (cdDocente);
/* FK for reference to Titulacao through relation Tit_doc_tit */
ALTER TABLE TitulacaoDocente ADD
   CONSTRAINT Tit_doc_tit FOREIGN KEY (CDTITULACAO)
   REFERENCES Titlacao (cdTitulacao);
```

ANEXO 2 – PERGUNTAS UTILIZADAS NA AVALIAÇÃO DA FERRAMENTA

Características relacionadas ao processo de ciclo de vida

Desenvolvimento de diagrama

A ferramenta permite incluir, mover, redimensionar e excluir objetos?

A ferramenta permite agrupar, desagrupar e mover objetos agrupados?

A ferramenta reposiciona objetos ligados a um objeto quando o mesmo for movido?

A ferramenta suporta todos os diagramas da UML?

Todos os diagramas da ferramenta funcionam?

Análise de diagrama

A ferramenta faz uma análise dos diagramas procurando por erros?

A ferramenta indica quais os erros encontrados?

Prototipação

A ferramenta suporta a geração de protótipos de aplicações?

A ferramenta possui opções para geração de protótipos?

Geração de código

A ferramenta permite a geração de código fonte a partir do modelo entidade-relacionamento de dados criado?

A ferramenta permite a geração de scripts para gerar as tabelas, com base no modelo de dados criado?

Quais as linguagens suportadas pela ferramenta para a geração de scripts?

Geração das telas

A ferramenta possui opções para geração de telas?

Quantas linguagens são suportadas pela ferramenta de geração de telas?

A ferramenta permite escolher elementos de interface como menus, botões e barras?

Geração de relatórios

A ferramenta possui opções para a geração de relatórios?

Engenharia Reversa de Dados

A ferramenta permite a engenharia reversa de dados?

A ferramenta suporta a extração dos códigos para a engenharia reversa de quantas linguagens?

Extração Automática de Dados e Geração de Documentação

A ferramenta permite a geração de dados de acordo com as especificações do desenvolvedor?

A ferramenta permite configurar os tipos de relatórios que se deseja emitir?

Características relacionadas ao uso da ferramenta CASE

Características de Hardware requerido pela ferramenta

Qual o espaço em disco rígido necessário para a instalação da ferramenta?

Qual a quantidade de memória RAM recomendada para a execução da ferramenta?

Ambiente de software requerido pela ferramenta

A ferramenta pode ser executada em quantas plataformas (sistemas operacionais)?

Repositório de software (base de informação)

A ferramenta possui um repositório?

A ferramenta permite a criação de informações (tabelas, atributos e relacionamentos) através do repositório?

Compatibilidade com elementos do ambiente

A ferramenta possui quais habilidades de interoperabilidade?

Integração de dados

A ferramenta permite utilizar informações de outras ferramentas?

Ambiente de hardware e software dos produtos da ferramenta

A ferramenta suporta quantos ambientes de hardware para sua execução?

Tamanho de aplicação suportado

Existe um número máximo de linhas de código possível de ser gerado pela ferramenta ?

Existe um número máximo de entidades que podem ser criadas na ferramenta, por aplicação?

Linguagens suportadas

A ferramenta suporta geração de códigos para quantas linguagens?

Banco de dados suportados

A ferramenta suporta quantos bancos de dados?

Suporte metodológico

A ferramenta suporta quantas metodologias?

Características gerais relacionadas a qualidade

Segurança

A ferramenta possui mecanismos eficazes de proteção contra o uso não autorizado, ilegal ou inadequado?

A ferramenta permite algum tipo de padrão de modelagem ou norma conhecida?

Conformidade técnica

A ferramenta segue algum padrão de modelagem ou norma conhecida?

A ferramenta está disponível em ambiente multiplataforma, ou seja, é suportado pelos sistemas operacionais mais utilizados, tais como: Windows, Windows NT, OS/2 e Unix?

Integridade dos dados

A ferramenta apresenta confiabilidade no caso de haver a necessidade de se recuperar os dados?

Tolerância a falhas

A ferramenta possui mecanismos que previnam possíveis falhas decorrentes de hardware ou software?

Recuperabilidade

A ferramenta possui recurso ou ferramenta interna que permite recuperar dados que eventualmente algum tipo de dano?

A ferramenta oferece recurso para que o usuário desfaça os processos efetuados?

Inteligibilidade

A interface que a ferramenta apresenta está condizente com as habilidades do usuário para o qual ela foi projetada?

A ferramenta apresenta organização em sua interface de forma que as informações possam ser manipuladas sem maiores problemas?

Operacionalidade

A ferramenta é clara e concisa nos seus processos de forma que o usuário saiba realmente o que está acontecendo?

A ferramenta gera LOG ou histórico para os processos que foram executados?

Manipulação de erros

A ferramenta possui consistências eficazes de forma que o usuário não possa provocar erros e inconsistências?

Na ferramenta, as mensagens são claras de forma que o usuário saiba qual é o erro e como deve corrigi-lo?

A ferramenta ao encontrar um erro, aborta o processo mostrando a mensagem?

Facilidade de aprendizagem

A disposição dos elementos visuais da ferramenta facilita o entendimento do usuário quanto a sua utilização?

A documentação permite que o usuário possa aprender a ferramenta sozinho?

Os treinamentos oferecidos pelo fabricante da ferramenta são de fácil acesso?

Qualidade da documentação

A ferramenta possui manuais e help detalhado e esclarecedores sobre cada processo?

A ferramenta possui tutoriais ou exemplos auto-explicativos?

Facilidade de instalação

Quantos passos são necessários para a instalação da ferramenta?

Qual a facilidade da documentação que auxilia da instalação da ferramenta?

Qual o tempo para a instalação da ferramenta?

Tempo de resposta

A ferramenta possui um tempo de resposta apropriado?

Requisitos de armazenamento

Qual o espaço que a ferramenta ocupa em disco?

As bases geradas pela ferramenta em média são?

Capacidade de memória aceitável

Qual a capacidade mínima aconselhada pelo fabricante para o funcionamento da ferramenta?

Qual o desempenho da ferramenta em uma máquina com esta especificação de memória?

Portabilidade para diferentes plataformas de hardware

Em quais ambientes de hardware a ferramenta executa?

Compatibilidade entre diferentes sistemas operacionais

Para quais dos sistemas operacionais relacionados a versão avaliada disponibiliza uma versão?

Características gerais não relacionadas a qualidade

Perfil do fornecedor

A quanto tempo o fornecedor da ferramenta está no mercado?

O fornecedor desenvolve outros produtos além do avaliado?

Qual a abrangência de mercado do fornecedor?

O fornecedor da ferramenta está presente na região?

Perfil do produto

De que forma pode ser a aquisição da ferramenta?

A ferramenta pode ser atualizada via internet?

Existem versões de avaliação da ferramenta?

Disponibilidade de treinamento

O fornecedor oferece cursos de treinamento?

Acontecem periodicamente encontros ou simpósios sobre a ferramenta?

REFERÊNCIAS BIBLIOGRÁFICAS

ABNT - ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **ISO/IEC 14102:** Orientação para avaliação e seleção de ferramentas CASE. Rio de Janeiro, 2000.

ANDRADE, Francisco Edmundo. **Automação de sistemas da informação II**. Faculdade integrada da Upis, 2000. Disponível em: <http://www.upis.br/edmundo/asi_II/1999s2/notes/DefinicaoUML.html>. Acesso em: 05 mar. 2001.

AYROSO, Vanessa Dalú. **Desenvolvimento e implementação de um sistema de inventário utilizando ferramentas Oracle CASE**. 1993. 98 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

AZEVEDO, Manoel Santos. **Ferramenta CASE,** abr. [1998], Disponível em : <http://www.internext.com.br/mssa>. Acesso em: 27 mar. 2001.

BARROS, Pablo. **Linguagem de Modelagem Unificada**, dez. 1998. Disponível em: <http://cc.usu.edu/~slqz9/uml/>. Acesso em: 30 abr. 2001.

CANTÚ, Marco. **Dominando o Delphi 5** : A Bíblia. São Paulo: Makron Books, 2000.

CHIKOFSKY, Elliot. **Computer-Aided Software Engineering (CASE)**. COMPUTER IEEE Computer Society, 1993.

EYNG, Juliana; HERMIDA, Alexandre Campos; SILVA, Andréa Vergara. **Seminários de sistemas orientados a objetos –** Universidade Federal de Santa Catarina, 2000. Disponível em : http: <http://wwwedit.inf.ufsc.br:1194/2000/Links.html>. Acesso em : 27.04.2000.

FISCHER, Alan S. **CASE:** Utilização de ferramentas para desenvolvimento de software**.** Rio de Janeiro: Campus, 1990.

FOURNIER, Roger. **Guia prático para o desenvolvimento e manutenção de sistemas estruturados.** São Paulo: Makron Books, 1994.

FUGGETTA, Alfonso. **A classification of CASE technology.** COMPUTER IEEE Computer Society, dec. 1993.

FURLAN, José Davi. **Modelagem de objetos através da UML** – the unified modeling language. São Paulo: Makron Books, 1998.

LOH, Stanley.**Ambiente de desenvolvimento de software e ferramenta CASE,** mar. 1996, Disponível em : <http://atlas.ucpel.tche.br/~loh>. Acesso em: 27 mar. 2001.

MCCLURE, Carma. **CASE is software automation.** New Jersey : Prentice_hall, 1989.

MARTIN, James; McCLURE, Carma. **Software maintenance –** the problem and its solutions. New Jersey: Prentice_Hall, 1983.

MAFFEO, Bruno. **Engenharia de software e especificação de sistemas**. Rio de Janeiro: Campus, 1992.

MEC/SESU. Comissão de especialistas de ensino de Computação e Informática. Desenvolvido pelo MEC Secretaria de Educação Superior, 1997. Apresenta textos sobre temas de Políticas do ensino superior e processos de avaliação do ensino superior. Disponível em: <http://www.inf.ufrgs.br/mec/ceeinf.renovacao.html>. Acesso em: 27 mar. 2001.

NASCIMENTO, João Belmiro do. **Metodologia de desenvolvimento de sistemas**. São Paulo: Érica, 1993.

POPKIN SOFTWARE SYSTEMS INC. **System Architect user guide**. New York: Popkin, 1998.

PRESSMAN, Roger S. **Engenharia de software**. São Paulo: Makron Books, 1995.

RUMBAUGH, James et al. **Modelagem e projetos baseados em objetos.** Rio de Janeiro: Campus, 1994.

SILVA, Alberto; VIDEIRA, Carlos. **UML metodologias e ferramentas CASE**. Lisboa: Centro Atlântico Ltda, 2001.

NAU, Rodrigo Dadam. **Protótipo de uma ferramenta CASE para a fase de análise da proposta UML**. 1998. 73 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

WAZLAWICK, Raul Sidnei. **Sistemas orientados a objetos** – Universidade Federal de Santa Catarina, 2000. Disponível em : http: <http://wwwedit.inf.ufsc.br:1194/users/grupo1/index.htm>. Acesso em: 25.04.2001.

WEINRICH, Jair. **Software de apoio à avaliação e seleção de ferramentas CASE baseado na norma ISO/IEC 14102**. 1999. 76 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.