UNIVERSIDADE EDUARDO MONDLANE

FACULDADE DE ENGENHARIA

Curso de Engenharia Informática

Estudo sobre o desenvolvimento de ferramentas CASE como instrumentos para auxiliar o processo de ensino e aprendizagem no ensino superior em Moçambique

Autor:

Helber Chin Ku Chon Choo

Supervisor:

Eng. Gonçalves Justino Doho

Co-supervisores:

Dr. Valério João

Professor Dr. Marco Ronchetti

Maputo, Agosto de 2013

UNIVERSIDADE EDUARDO MONDLANE

FACULDADE DE ENGENHARIA

Curso de Engenharia Informática

Estudo sobre o desenvolvimento de ferramentas CASE como instrumentos para auxiliar o processo de ensino e aprendizagem no ensino superior em Moçambique

Autor:

Helber Chin Ku Chon Choo

Supervisor:

Eng. Gonçalves Justino Doho

Co-supervisores:

Dr. Valério João

Professor Dr. Marco Ronchetti

Maputo, Agosto de 2013

TERMO DE ENTREGA DO RELATÓRIO

UNIVERSIDADE EDUARDO MONDLANE

FACULDADE DE ENGENHARIA

DEPARTAMENTO DE ENGENHARIA ELECTROTECNICA

TERMO DE ENTREGA DE RELATÓRIO DO TRABALHO DE LICENCIATURA

Declaro que o estudante _____ entregou no dia __/__/2013 as ____ copias do relatório do seu Trabalho de Licenciatura com a referencia:_____

intitulado: Estudo sobre o desenvolvimento de ferramentas CASE como instrumentos para auxiliar o processo de ensino e aprendizagem no ensino superior em Moçambique

Maputo, ____ de _____ de 2013

Assinatura

_____

(Chefe da secretaria do DEEL)

Dedico este trabalho a minha família, amigos, colegas e professores que sempre me apoiaram e deram me força sempre que precisa-se.

"As mais belas obras de arte surgem dos sentimentos expressos pelo artista e não pela sua sede de perfeccionismo" (Helber Chin Ku Chon Choo)

Agradecimentos

Expressos os meus sinceros agradecimentos a todos aqueles que directa ou indirectamente, consciente ou inconscientemente, contribuíram para a concretização deste trabalho.

Á minha família por me ter ajudado durante todo esse tempo, dando coragem para nunca desistir.

Aos meus supervisores, por me terem indicado as directrizes para a realização deste trabalho e principalmente por terem esclarecido de forma paciente todas as preocupações por mim apresentadas.

Aos meus amigos que sempre deram me força para continuar e mostrar o meu potencial.

Resumo

Vivemos actualmente numa era onde a informática domina grande parte do mercado mundial, sendo os programas de computador (*Software*), uma das ferramentas mais importantes para a maioria das organizações em todo o mundo, possibilitando que as mesmas logrem realizar as suas actividades com maior eficácia e eficiência. Com as últimas inovações tecnologias e abertura do mercado mundial, o processo de desenvolvimento de *Software* tornou-se mais exigente requerendo integrantes que possuam conhecimentos sólidos nas diferentes áreas de desenvolvimento de *Software*.

Perante esta situação, instituições que ensinam o desenvolvimento de *Software* adoptam a utilização de programas que auxiliam o processo de desenvolvimento (mais conhecidos por ferramentas CASE) durante o processo de ensino-aprendizagem. Estas ferramentas possibilitaram aos professores e estudantes desenvolverem projectos práticos e discutirem programas grandes e mais complexos durante as suas aulas. Entretanto, apesar destas vantagens, as ferramentas CASE geralmente são projectadas mormente para uso comercial e não sendo muito adequados para uso educacional, requerendo assim uma curva de aprendizagem bastante acentuada por parte dos estudantes para que possam usá-las efectivamente.

Neste contexto, o presente trabalho visa fazer um estudo de desenvolvimento de ferramentas CASE capazes de auxiliar o processo de ensino-aprendizagem nas instituições de ensino superior em Moçambique.

Palavras-chaves: Ferramentas CASE educacionais, TIC, Computador na Educação, desenvolvimento de Software educacional.

Abstract

We live in an era where information technology dominates much of the world market, and computer programs are among the most important tools used in organizations around the world, enabling them to carry out their activities more effectively and efficiently. With the new technological innovations and globalization, the process for developing software has become more complex requiring people with substantial knowledge in multiple areas of software development.

Given the situation, the institutions that teach software development adopt the use of programs that help the software development process (commonly known as CASE tools) during the teaching and learning process. These tools enable teachers and students to develop practical projects and discuss large and complex programs during their lessons. However, despite these advantages, CASE tools are usually designed mainly for commercial use and not very suitable for educational use, thus requiring a steep learning curve from the students so they can use them effectively.

In this context, this work aims to study the development of CASE tools that can assist the process of teaching and learning in higher education institutions in Mozambique.

Key-words: Educational CASE tools, ICT, computer in education, development of educational Software.

Índice

Lista de Abreviaturas e Símbolos

API	Interface de Programação de Aplicações
CASE	Computador Auxilia Engenharia de Software
GUI	Interface Gráfica do Utilizador
IDE	Ambiente de Desenvolvimento Integrado
ISUTC	Instituto Superior de Transportes e Comunicação
PBD	Projecto de base de dados
POO	Programação Orientada a Objectos
RDBM	Modelador de Base de Dados Relacional
SGBD	Sistema de gestão de base de dados
SQL	Linguagem de Consulta Estruturada
UEM	Universidade Eduardo Mondlane
USTM	Universidade São Tomãs de Moçambique

Glossário de termos

API – A Interface de Programação de Aplicativos (do Inglês *Application Programming Interface*) é um conjunto de rotinas e padrões estabelecidos por um programa de computador para a utilização das suas funcionalidades por aplicativos que não pretendem envolver-se em detalhes da implementação do programa, mas apenas utilizar as suas funcionalidades.

Ferramenta CASE – Ferramentas CASE são programas utilizados para suporte das actividades do processo de engenharia de *Software*.

IDE – Ambiente de Integrado de Desenvolvimento (do Inglês *Integrated Development Environment*) é um programa de computador que reúne características e ferramentas de apoio ao desenvolvimento de *Software* com o objectivo de agilizar este processo.

Parcimónia - Acção ou costume de economizar.

Sociedade de conhecimento – é compreendida como aquela na qual o conhecimento é o principal factor estratégico de riqueza e poder, tanto para as organizações quanto para os países. Nesta sociedade, a inovação tecnológica ou novo conhecimento, passa a ser um factor importante para a produtividade e para o desenvolvimento económico dos países.

Lista de figuras

Lista de tabelas

Capitulo I – Introdução

1.1 Contextualização

Actualmente a informática domina grande parte do mercado mundial, sendo os programas de computador (*Software*), uma das ferramentas mais importantes para a maioria das organizações em todo o mundo, possibilitando que as mesmas logrem realizar as suas actividades com maior eficácia e eficiência. Estudos realizados por Filho W. P. (2001), Baab (2004), Pereira T. S. (2008), Teixeira & Brandão (2003), Silva (2007), Vannucci & Colla (2010) e Valente (1995) (1999), demostram que a utilização de *Software* educacional no processo de ensino- aprendizagem pode impactar de positivamente na aprendizagem do estudante, suavizando sua curva de aprendizagem. O *Software* educacional refere-se ao *Software* que pode ser usado por professores e alunos para apoiar o ensino e aprendizagem (National Centre for Technology in Education 2007).

Para Valente (1999), a informática poderá ser usada para suportar a realização de uma pedagogia que proporcione a formação dos alunos, possibilitando o desenvolvimento de habilidades que serão fundamentais na sociedade do conhecimento[1]. O uso do computador na educação deve ser considerado como meio e ampliação das funções do professor, favorecendo mudanças nas condições e no processo de ensino-aprendizagem (Silva 2007). Como forma de capacitar os seus estudantes com conhecimentos sólidos na área de desenvolvimento de *Software*, algumas instituições adoptam a utilização de *Software* educacionais, *Software* capazes de auxiliar o processo de desenvolvimento (mais conhecidos por ferramentas CASE) durante o processo de ensino.

Entretanto o *Software*, de modo genérico, é uma entidade que se encontra em quase constante estado de mudança, as mudanças ocorrem por necessidade de corrigir erros existentes no *Software* ou de adicionar novos recursos e funcionalidades (A. M. Filho 2007), sendo que engenharia de *Software* vem desempenhando um papel muito importante no desenvolvimento de programas de computador, desde a elicitação aos testes, o processo de *Software* sofreu diversos aprimoramentos e hoje é compreendido como essencial para a criação de qualquer sistema computacional (Farias 2001). O

[1]A sociedade de conhecimento é compreendida como aquela na qual o conhecimento é o principal factor estratégico de riqueza e poder, tanto para as organizações quanto para os países. Nesta sociedade, a inovação tecnológica ou novo conhecimento, passa a ser um factor importante para a produtividade e para o desenvolvimento económico dos países.

processo de desenvolvimento de *Software* inclui todas as actividades envolvidas no desenvolvimento de *Software* (Sommerville 2011).

As ferramentas CASE possibilitam aos professores e estudantes desenvolverem projectos práticos e discutirem sobre problemas mais complexos durante as suas aulas. Entretanto, apesar destas vantagens, as ferramentas CASE geralmente são projectadas mormente para uso comercial e não sendo muito adequados para uso educacional, requerendo assim uma curva de aprendizagem bastante acentuada por parte dos estudantes para que possam usá-las efectivamente. O desenvolvimento de programas de computador educacionais é uma tarefa difícil, pois envolve muitos factores interagindo mutuamente e com a cooperação de várias figuras profissionais (Vannucci e Colla 2010). O processo de desenvolvimento de *Software* educacional deve levar em consideração ambos os factores pedagógicos e da ciência de computação que são normalmente tratados por duas pessoas ou grupos distintos (Valente 1995).

Nesse âmbito, o presente trabalho de licenciatura visa estudar uma metodologia para o desenvolvimento de ferramentas CASE educacionais.

1.2 Motivação

Nas instituições de ensino superior de Moçambique, ainda não é notável uma grande adopção da utilização de ferramentas CASE como instrumentos de ensino e aprendizagem, sendo que em algumas cadeiras é notável a importância da utilização dessas ferramentas. Nos últimos dois anos, em conversas informais com estudantes e docentes dos cursos de relacionados com ciências de computação da Universidade Eduardo Mondlane (UEM), da Universidade Pedagógica de Moçambique e do Instituto Superior de Transportes e Comunicações (ISUTC), o autor constatou que as instituições de ensino superior em Moçambique que leccionam cursos relacionados com ciências de computação, utilizam pelo menos uma ferramenta CASE para aulas de programação, contudo para aulas envolvendo análise de requisitos, desenho e testes de sistemas, grande parte das instituições não utilizam ferramentas CASE, geralmente porque as ferramentas existentes para o auxilio destas actividades serem desenvolvidas para uso comercial ou serem muito complexas ao ponto de requererem que os estudantes já estejam dotados de fortes conhecimentos básicos na área que elas suportam, e em alguns casos as ferramentas não são utilizados por não serem

consideras como instrumentos de ensino e aprendizagem por parte de alguns professores.

Neste contexto, o presente trabalho visa fazer um estudo de desenvolvimento de ferramentas CASE capazes de auxiliar o processo de ensino-aprendizagem nas instituições de ensino superior em Moçambique, observando as teorias de aprendizagem e analisando as características das ferramentas educacionais. Com os resultados deste estudo, propor-se-á um conjunto de requisitos mínimos para ferramentas CASE mais adequadas ao uso no processo de ensino-aprendizagem.

1.3 Objectivos do trabalho

1.3.1 Objectivo geral

Fazer um estudo do desenvolvimento de programas de computador capazes de auxiliar o processo de ensino e aprendizagem no desenvolvimento de *Software* em instituições de ensino superior em Moçambique

1.3.2 Objectivos específicos

O presente trabalho tem como objectivos específicos os seguintes:

- Fazer um estudo comparativo das teorias e tecnologias utilizadas nos principais programas de computador utilizados no processo de ensino e aprendizagem nas instituições de ensino superior;
- Fazer uma revisão das teorias de aprendizagem auxiliadas por computador;
- Identificar os requisitos de ferramentas CASE para diferentes áreas de aprendizagem;
- Produzir um modelo de desenvolvimento de ferramentas CASE educacionais;
- Desenvolver um protótipo baseado no modelo produzido;
- Discutir resultados, emitir conclusões e recomendações.

1.4 Metodologia

1.4.1 Metodologia para desenvolvimento do trabalho

No presente trabalho foram utilizadas as seguintes metodologias de investigação a fim de alcançar os objectivos do projecto:

- **Consultas regulares aos supervisores** – como forma de manter um maior controlo sobre as actividades realizadas durante a realização do trabalho, foram feitas consultas com intervalo máximo de catorze (14) dias aos supervisores;
- **Pesquisa e revisão bibliográfica** – É uma etapa fundamental em todo trabalho científico que influenciará todas as etapas de uma pesquisa (Amaral 2007). Foi realizada uma pesquisa bibliográfica exaustiva sobre o tema em questão e depois fez se a colecta de dados. O acesso a bibliografia foi feito manualmente e electronicamente;
    - **Manualmente** – Consistiu em pesquisar directamente nos livros e artigos de referência existentes na biblioteca da universidade de Trento. Foram observados livros e artigos como: Elmasri e Navathe (2003), Fuggett (1993), Sommerville (2011) e Vannucci e Colla (2010).
    - **Electronicamente** – Consistiu em pesquisar por informação em *sites* credíveis, tais como: *IEEE Xplore Digital Library* (http://ieeexplore.ieee.org) e a *ACM digital library* (http://dl.acm.org).
- **Sistémica** – Foi estabelecido um conjunto de passos sistematizados que possibilitaram aplicar o pensamento sistémico de maneira organizada, de modo que ao final de cada etapa, obteve se resultados que serviram de entrada no passo subsequente. Esta abordagem foi utilizada para manter a direcção dos objectivos previamente traçados;

1.4.2 Metodologia para desenvolvimento do protótipo

A fim de testar a metodologia proposta para desenvolvimento de ferramentas CASE educacionais, desenvolveu-se um protótipo seguindo a metodologia proposta. O protótipo consistiu em desenvolver uma ferramenta CASE para auxiliar o processo de ensino-aprendizagem nas aulas de base de dados da faculdade de engenharia da UEM, contando com a participação do professor de base de dados da faculdade de engenharia da UEM, o professor Vali Issufo e com a participação do professor Marco Ronchetti.

Uma vez que o protótipo desenvolvido ainda não foi implementado, no presente trabalho apresenta-se somente as fases de análise de requisitos, desenho da arquitectura e desenho da interface do utilizador.

1.5 Conteúdo e organização do trabalho

O presente trabalho está dividido em sete (7) partes:

- **Capitulo I – Introdução**: Este é o primeiro capítulo do trabalho, nele são abordados aspectos introdutórios, apresentando-se a importância e objectivos do trabalho, fornecendo detalhes sobre sua elaboração e a forma em que as pesquisas foram realizadas.

- **Capitulo II – Ferramentas CASE no processo de ensino-aprendizagem:** Neste capítulo é feita a descrição teórica dos aspectos técnicos abordados durante a realização do trabalho, apresentando-se e reflectindo sobre diferentes pontos de vistas de vários autores. Nele é abordado conceitos básicos sobre ferramentas CASE, metodologias de desenvolvimento de *Software* e algumas teorias sobre o processo de ensino- aprendizagem.

- **Capitulo III** – **Ferramentas CASE educacionais**: Neste capítulo são feitos estudos sobre as ferramentas CASE educacionais, onde são analisados os requisitos e especificações que as mesmas devem ter. Ainda neste capítulo propõe-se uma metodologia para desenvolvimento das ferramentas CASE educacionais.

- **Capitulo IV – Caso de Estudo**: Neste capítulo apresenta-se o desenvolvimento do protótipo de uma ferramenta CASE educacional para auxiliar o processo de ensino-aprendizagem na modelagem de bases de dados relacionais na Faculdade de Engenharia da Universidade Eduardo Mondlane, seguindo a metodologia de desenvolvimento proposta e tomando em consideração os requisitos e especificações propostas para ferramentas CASE educacionais.

- **Capitulo V – Discussão dos Resultados**: Neste capítulo são apresentados os resultados obtidos durante o desenvolvimento do protótipo e os seus méritos e limitações.

- **Capitulo VI – Conclusões e recomendações**: Nesta parte do trabalho, são apresentadas conclusões sobre o estudo realizado e recomendações para trabalhos futuros.

- **Bibliografia**: Nesta parte do trabalho, são referenciadas as obras que foram consultadas para o desenvolvimento do trabalho.

Capitulo II – Ferramentas CASE no processo de ensino-aprendizagem

2.1 Ferramenta CASE

Desde de o início dos anos 80 foram realizados muitos estudos relacionados a ferramentas CASE, levando a muitos pontos de vista diferentes sobre as ferramentas CASE. Para Orlikowski (1989), as ferramentas CASE são programas de *Software* que automatizam ou apoiam as tarefas que constituem normalmente práticas de desenvolvimento de sistemas de informação. Krishnamurthy (2001) defende que as ferramentas CASE são ferramentas que fornecem assistência automatizada para desenvolvimento de *Software*, e para Sommerville (2011) as ferramentas CASE são programas utilizados para suporte das actividades do processo de engenharia de *Software*. Apesar de diferentes todas essas definições convergem no facto de aceitarem que as ferramentas CASE auxiliam o processo de desenvolvimento de *Software*.

As ferramentas CASE apresentam-se úteis no auxílio de um projecto de Software (Rader, Brown e Morris 1993), pois apoiam um ou mais processos de desenvolvimento de *Software*, seu uso tem um impacto directo no tempo e custo para o desenvolvimento de *Software* e, além disso, a qualidade dos produtos desenvolvidos. De acordo com (Krishnamurthy 2001) o interesse em ferramentas CASE é baseado em expectativas sobre o aumento da produtividade, melhoria da qualidade do produto, facilitando a manutenção e tornando a tarefa dos engenheiros de *Software* menos odioso e mais agradável.

2.1.1 Classificação de ferramentas CASE

As ferramentas CASE podem ser classificadas de várias formas diferentes, dependendo da abordagem utilizada, no presente trabalho, as ferramentas CASE foram classificadas de duas formas, de acordo com as actividades que suportam e de acordo com a tecnologia do processo de produção.

2.1.1.1 Actividades suportadas

De acordo com as actividades que suportam, as ferramentas CASE são classificadas em três categorias distintas:

- **Ferramentas *Upper* CASE** – As ferramentas de lidam com os níveis de abstracção mais elevados, isso implica a fase de requisitos, análise e desenho (Oliveira, Murta e Werner 2005).
- **Ferramentas *Lower* CASE** – Apoias as actividades das fases decodificação, testes e implementação. As ferramentas *lower* CASE processam os dados das ferramentas *Upper* CASE e fornecem dados de saída para os desenvolvedores. (AAFRIN 2011)
- **Ferramentas *Integrated* CASE (I-CASE)** – Também são conhecidas como ferramentas que atravessam o ciclo de vida (*Cross Life-Cycle tools*). A ferramenta actua como uma ponte que conecta as *upper* CASE e as *lower* CASE. Estas ferramentas apresentam mecanismo de partilha de informação com todas as ferramentas existentes no ambiente de desenvolvimento.

2.1.1.2 Tecnologia do processo de produção

Fuggett (1993) baseando-se em uma plataforma geral para o processo de produção de *Software*, apresentou um modelo para classificar ferramentas CASE de acordo com a tecnologia do processo de produção, do qual divide-as em três níveis de granularidade:

- **Ferramenta** – Suporta somente actividades especificas.
- **Ferramenta CASE *workbenches*** – Suporta somente uma ou mais actividades do processo de desenvolvimento de *Software*.
- **Ferramenta CASE ambiental** – Suporta uma grande parte do processo de desenvolvimento de *Software*.

Além da classificação de Fuggett, existem outras classificações com diferentes níveis de granularidade, como a de Sommerville (2011) que considera somente dois níveis (ferramenta e ambiente) e a de Fernstrõm, Nãrfelt e Ohlsson (1992) que considera quatro níveis (serviço, ferramenta, serie de ferramentas e ambiente). No entanto todas essas classificações acabam por ser envolvidas nas classes dos três níveis de granularidade definidas por Fugget (Tabela 2-1).

Nível de granularidade	Classes
Ferramentas	Ferramentas de edição, programação, verificação e validação, gerenciamento de configuração, métricas e mensuração, gerenciamento de projectos e diversas.
CASE *workbenches*	Planeamento de negócios e modelagem, análise e desenho, desenvolvimento de interface do utilizador, programação, verificação e validação, manutenção e engenharia reversa, gerenciamento de configuração, gerenciamento de projectos
CASE ambiente	*Toolkits*, língua-centrado, integrado, quarta geração, processo-centrado.

Tabela 2-1: Classificação de ferramentas CASE – Classes de Ferramentas CASE

2.2 O computador no processo de ensino-aprendizagem

A introdução do computador na educação tem provocado uma verdadeira revolução na nossa concepção de ensino e aprendizagem (Valente 1995). De acordo com Valente (1995), os computadores podem ser utilizados para ensinar e mostram-se como uma versão computadorizada das metodologias de ensino.

2.2.1 Teorias de aprendizagem convencionais

Uma teoria de aprendizagem é uma construção humana para interpretar sistematicamente a área de conhecimento que chamamos aprendizagem (Moreira 2011). Aprendizagem é uma característica inerente a todos os seres que raciocinam (Santos 2006). Existem diversas teorias como formas explicativas da aprendizagem, as quais podem ser agrupadas em três (3) abordagens: a comportamentalista (behaviorismo), a cognitivista (construtivismo) e a humanista.

- **Filosofia Comportamentalista** – A abordagem comportamentalista analisa o processo de aprendizagem, desconsiderando os aspectos internos que ocorrem na mente do aprendiz, centrando-se no comportamento observável (Santos 2006), ou seja, a abordagem comportamentalista considera somente os aspectos comportamentais do aprendiz, desconsiderando os aspectos mentais.
- **Filosofia Cognitivista ou construtivismo** – A filosofia cognitivista trata dos processos mentais, se ocupa da atribuição de significados, da compreensão, transformação, armazenamento e uso da informação envolvida na cognição

(Moreira 2011). No construtivismo, o aprendiz deixa de ser visto como receptor de conhecimentos e passa a ser considerado agente da construção de sua estrutura cognitiva (Sobrinho 2012).

- **Teoria Humanista** – A abordagem humanista prioriza como base fulcral da aprendizagem a auto-realização do aprendiz, havendo uma valorização tanto do aspecto cognitivo, quanto do motor como do afectivo (Santos 2006).

2.2.2 Computador na educação

De acordo com Valente (1999), os computadores deverão assumir duplo papel na educação, como ferramentas de comunicação e como máquinas de ensinar. Como máquinas de ensinar eles podem ser categorizados da seguinte forma:

- **Programas tutoriais** – Nesta categoria a informação é organizada de acordo com uma sequência pedagógica particular e apresentada ao estudante, seguindo essa sequência, ou então o aprendiz pode escolher a informação que desejar (Valente 1999).
- **Programas de exercício-e-pratica** – Tipicamente os programas de exercício-e-prática são utilizados para fazer revisão de material já visto em aula principalmente, material que envolve memorização e repetição, como aritmética e vocabulário (Valente 1995).
- **Jogos educacionais** – Os jogos educacionais podem ser vistos como tutoriais ou Software de simulação aberta, dependendo do quanto o aprendiz pode descrever suas ideias para o computador.
- **Simulação** – Os programas para simulação possibilitam simular um determinado fenómeno no computador. As simulações podem ser fechadas ou abertas.
    - **Simulações fechadas** – Os fenómenos são previamente implementados no computador e o aprendiz pode alterar o valor de alguns parâmetros a fim de assistir o desenrolar desse fenómeno.
    - **Simulações abertas** – A simulação aberta fornece algumas situações já previamente definidas e outras que devem ser definidas pelo aprendiz. Nesta simulação o aprendiz é encorajado a descrever ou implementar alguns aspectos do fenómeno.

- **Modelagem –** Os programas de modelagem são os que possibilitam ao aprendiz, escolher um fenómeno a modelar, desenvolver o seu modelo e implementá-lo no computador.

2.2.3 Ferramentas CASE no Ensino Superior em Moçambique

Todas as instituições de ensino superior que leccionam cursos relacionados com ciências de computação em Moçambique, utilizam no mínimo uma ferramenta CASE para aulas de programação ou base de dados, contudo para aulas envolvendo desenho, análise ou concepção de sistemas, grande parte das instituições não utilizam ferramentas CASE. Geralmente, o professor desenha os diagramas/modelos e suas componentes em um quadro e os estudantes desenvolvem os seus projectos em folhas de papel.

Por exemplo, nas aulas de base de dados, do Instituto Superior de Transportes e Comunicação (ISUTC) e na Universidade São Tomãs de Moçambique (USTM), não utilizam nenhuma ferramenta CASE para concepção, desenho e análise de base de dados durante as suas aulas. A Universidade Eduardo Mondlane (UEM) começou a utilizar em 2012 uma ferramenta CASE (MySQL *Workbench*) para desenhar bases de dados, contudo o MySQL *Workbench* não segue a notação utilizada durante as aulas de base de dados na UEM.

2.3 Metodologias de desenvolvimento de Software

Um processo de *Software* é um conjunto de actividades relacionadas, que conduz à produção de um produto de *Software*. Essas actividades podem envolver o desenvolvimento de *Software* a partir do zero ou fazer algumas modificações em um já existente. O modelo de processo de *Software* é uma representação simplificada de um processo de *Software* (Sommerville 2011).

2.3.1 Desenvolvimento em cascata

O modelo cascata leva as actividades fundamentais do processo de especificação, desenvolvimento, validação e evolução e as representa como fases distintas do processo (Sommerville 2011). Esta metodologia promove o desenvolvimento de projectos e requisitos bem definidos (Costa, Loureiro e Reis 2009).

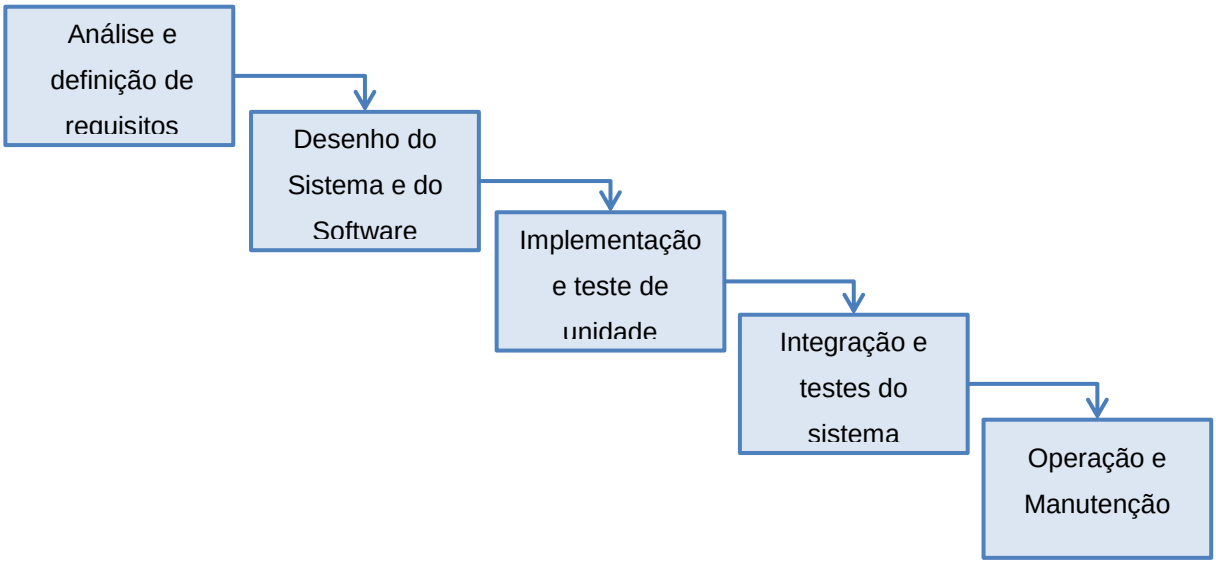


Figura 2-1: Modelo de desenvolvimento de *Software* em cascata

2.3.1.1 Benefícios do modelo cascata

- As fases são processadas e completadas uma de cada vez;
- É um modelo simples de implementar e fácil de utilizar;
- Fácil de gerir devido à rigidez do modelo, cada fase tem produtos específicos e a um processo de revisão;
- Funciona bem para pequenos projectos onde os requisitos são muito bem compreendidos;

2.3.1.2 Dificuldades do modelo cascata

- A versão funcional só é produzida no final do ciclo de vida de desenvolvimento;
- Grandes quantidades de risco e incerteza;
- Modelo pobre para projectos de longos e contínuos;
- Não é adequado para os projectos onde os requisitos estão em um risco moderado a alto de mudarem;
- Não é um bom modelo para projectos complexos e orientada a objectos;
- Quando uma falha é detectada na fase de testes, é muito difícil voltar atrás e rectificar algum erro na fase de concepção;

2.3.1.3 Quando utilizar o modelo cascata

- Definição do produto é estável;
- Há disponibilidade de recursos e conjunto de habilidades necessárias;
- O projecto é curto;
- Não há requisitos ambíguos;

- Requisitos muito bem conhecidos, claros e fixos;
- Tecnologia é compreendida;

2.3.2 Desenvolvimento incremental

Esta é uma abordagem para o desenvolvimento de *Software*, onde alguns dos incrementos desenvolvidos são entregues ao cliente e implementados para uso em um ambiente operacional (Sommerville 2011).

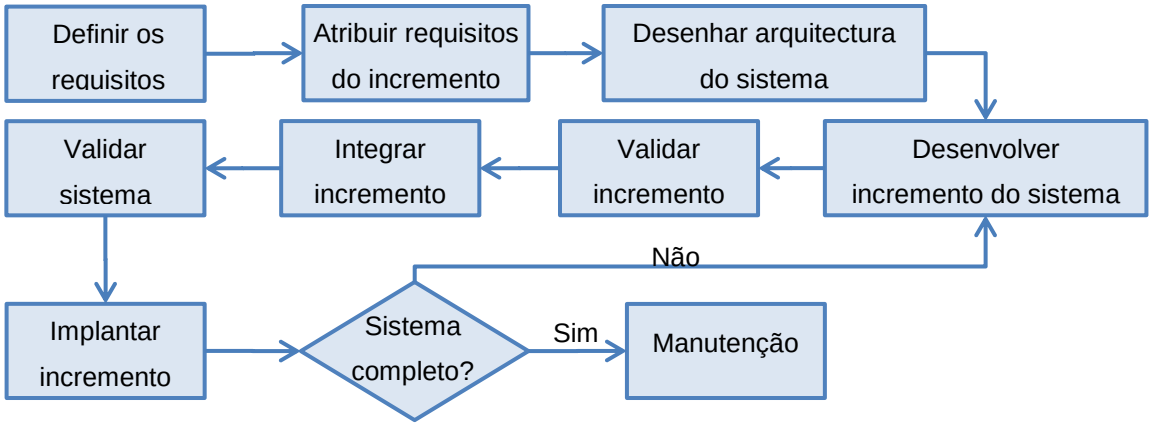


Figura 2-2: Modelo de desenvolvimento de *Software* incremental

2.3.2.1 Benefícios do modelo incremental

- Fácil de gerir o risco, pois estes são identificados e controlados (o seu impacto é reduzido) durante a sua iteração;
- Fácil de testar e depurar durante uma iteração menor;
- Flexível, menos custoso para alterar requisitos;
- Gera *Software* funcional rapidamente e no início do ciclo de vida do *Software*;
- O cliente pode responder a cada iteração;
- Reduz o custo de entrega inicial;

2.3.2.2 Dificuldades do modelo incremental

- O custo total é superior à do modelo cascata;
- Requer um bom planeamento e desenho;
- Requer uma definição clara e completa de todo o sistema antes que possa ser dividido e construído de forma incremental;

2.3.2.3 Quando utilizar o modelo incremental

- Existem algumas funcionalidades e metas de alto risco;
- Há uma necessidade de obter um produto funcional logo de início;

- Não há disponibilidade de recursos e conjunto de habilidades necessárias;
- Requisitos do sistema completo são claramente definidos e compreendidos;
- Requisitos principais devem ser definidos;
- Uma nova tecnologia está sendo usada;

2.3.3 Desenvolvimento em prototipagem

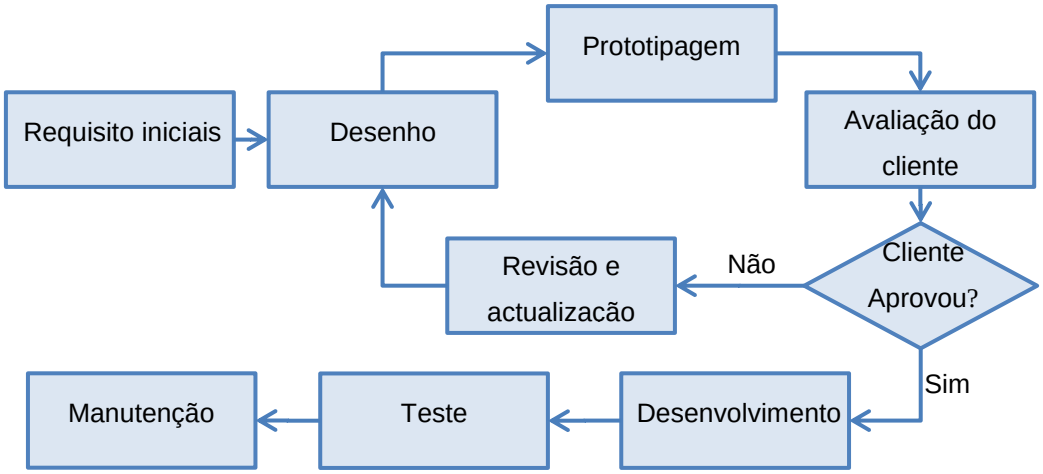


Figura 2-3: Modelo de desenvolvimento de *Software* em prototipagem

A metodologia de protótipos é uma forma de experimentação iterativa com o objectivo de obter informação para o processo de desenvolvimento (Costa, Loureiro e Reis 2009). Metodologias baseadas em Prototipagem realizaram as fases de análise, projecto e implementação simultaneamente. Todas essas fases são realizadas repetidamente num ciclo até que o sistema seja completado.

2.3.3.1 Benefícios do modelo de prototipagem

- Erros podem ser detectados cedo;
- Funcionalidade ausente pode ser identificada facilmente;
- Implementação rápida de uma aplicação funcional, apesar de incompleta;
- Os utilizadores estão activamente envolvidos no desenvolvimento;
- Os utilizadores adquirem maior compreensão do sistema que está a ser desenvolvido;
- Rápido *feedback* do utilizador, ajudando na obtenção das melhores soluções;

2.3.3.2 Dificuldades do modelo de prototipagem

- Análise do problema incompleta ou inadequada;
- Leva a uma forma de desenvolvimento de implementar e em seguida reparar;

- Na prática, pode aumentar a complexidade do sistema uma vez que o escopo do sistema pode expandir para além dos planos originais;

2.3.3.3 Quando utilizar o modelo de prototipagem

- Necessidade de desenhar-se uma boa interface de utilizador;
- O sistema desejado precisa ter muitas de interacção com os utilizadores finais;

2.3.4 Desenvolvimento em espiral

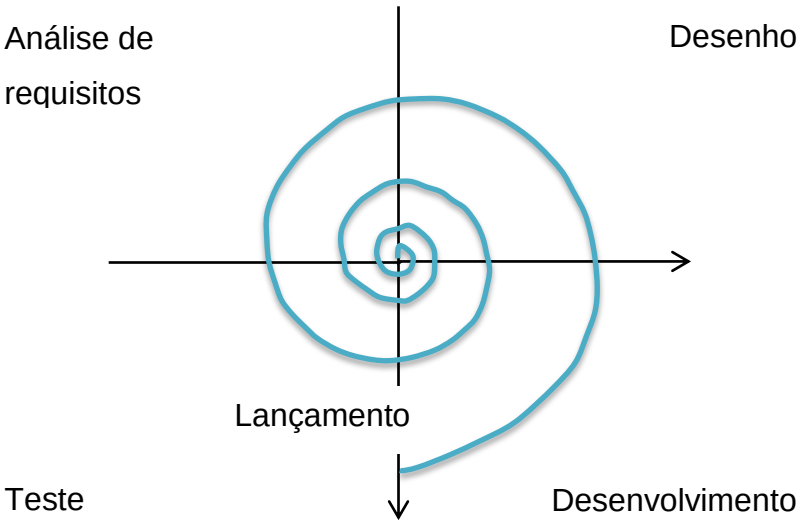


Figura 2-4: Modelo de desenvolvimento de *Software* em espiral

É um modelo de evolução, o qual combina a natureza iterativa da prototipagem com características da metodologia cascata (Costa, Loureiro e Reis 2009).

2.3.4.1 Benefícios do modelo espiral

- Aprovação forte e controle de documentação;
- Bom para projectos de grande porte e críticos;
- Maior previsão de risco devido a elevada quantidade de análise de risco;
- O *Software* é produzido no início do ciclo de vida do *Software*.

2.3.4.2 Dificuldades do modelo espiral

- Não funciona bem para projectos menores;
- Pode ser dispendioso de utilizar;
- Requer conhecimentos altamente específicos para a análise de risco;
- Sucesso do projecto é altamente dependente da fase de análise de risco;

2.3.4.3 Quando utilizar o modelo espiral

- Custos e avaliação de risco são importantes;

- Mudanças significativas são esperadas (pesquisa e exploração);
- Nova linha de produto;
- Os utilizadores não têm certeza de suas necessidades;
- Projectos de alto risco;
- Requisitos são complexos;

2.3.5 Desenvolvimento ágil

Desenvolvimento ágil representa um grupo de metodologias de engenharia de *Software* que prometem oferecer maior produtividade, qualidade e taxa global de sucesso no projecto, em projectos de desenvolvimento de *Software* (Ionel 2009). O desenvolvimento ágil é uma alternativa à gestão tradicional de projectos.

Algumas das metodologias ágeis são SCRUM (Schwaber e Beedle 2001) e programação estrema ou XP (Beck e Andres 2004). Os parâmetros para o desenvolvimento ágil, foram estabelecidos pelo Manifesto Ágil (Vide Anexo A1) em 2001 (Beck, Beedle, et al. 2001).

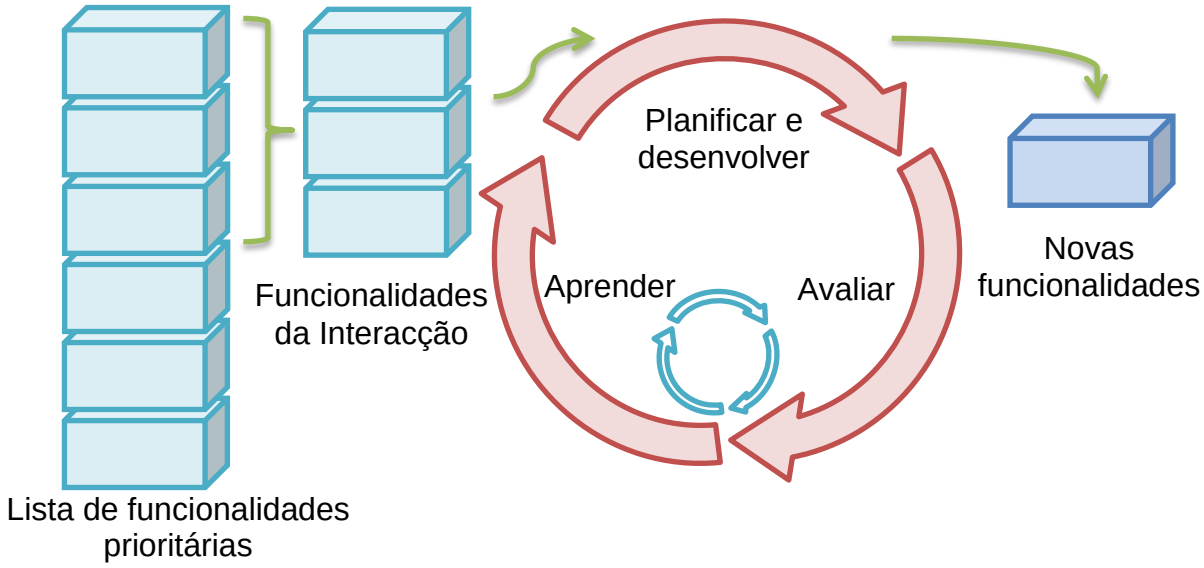


Figura 2-5: Visão geral das metodologias de desenvolvimento ágeis

2.3.5.1 Características das metodologias de desenvolvimento ágil

As metodologias de desenvolvimento ágil usualmente aplicam desenvolvimento iterativo e evolutivo de tempo limitado, empregam planeamento adaptativo, promovem entrega incremental e incluem outros valores que encorajam agilidade, resposta rápida e flexível à modificação (Larman 2007). Miller (2001) deu as seguintes características para o desenvolvimento ágil de Software:

- Modularidade no processo de desenvolvimento;
- Interactividade dos pequenos ciclos de desenvolvimento a fim de refinar o resultado;
- Intervalo de tempo de duas a seis semanas entre cada interacção;
- Parcimónia no processo de desenvolvimento, eliminando todas actividades desnecessárias;
- Adaptação com todos os possíveis riscos;
- Uma abordagem incremental que possibilita a construção de um sistema funcional com pequenos passos;
- Uma abordagem convergente, atacando todos os riscos e entregando o projecto em incrementos;
- Orientado para as pessoas;
- Modelo de trabalho colaborativo;

2.3.5.2 Benefícios do modelo ágil

- Adaptação regular à evolução e alteração dos requisitos;
- Comunicação presencial;
- Maior ênfase às pessoas do que às tecnologias;
- Satisfação do cliente através da entrega rápida e contínua de *Software* útil;

2.3.5.3 Dificuldades do modelo ágil

- Em alguns casos é difícil avaliar o esforço necessário para o início do ciclo de vida do desenvolvimento de *Software*, principalmente em grandes projectos;
- Há falta de ênfase no desenho e documentação necessária;
- Possibilidade de desvio dos objectivos iniciais, caso os representantes do cliente não conheçam exactamente o resultado desejado;
- Programadores iniciantes não são capazes de tomar as decisões necessárias durante o processo de desenvolvimento;

2.3.5.4 Quando utilizar o modelo ágil

- Alteração frequente dos recursos;
- Existe disponibilidade entre ambas as partes, desenvolvedores e os utilizadores finais;
- Há necessidade de implementar novas alterações;

Capitulo III – Ferramentas CASE educacionais

3.1 Análise de requisitos das ferramentas CASE educacionais

Baseando-se nas diferentes categorias de um *Software* educacional apresentado por Valente (1999), ao utilizar uma ferramenta CASE educacional o estudante passaria por diversas acções que acontecem em termos do ciclo **descrição – execução – reflexão – depuração – descrição**. A descrição seria o processo que ocorre no momento em que o estudante, utilizando os seus conhecimentos representa com auxílio da ferramenta os passos para a solução do problema. A execução é o processo em que o computador executa a informação recebida do estudante, fornecendo um *feedback* fiel e imediato, desprovido de qualquer animosidade ou afectividade que possa haver entre o estudante e computador (Valente 1999), na reflexão, o estudante deduz algum conhecimento através das acções ocorridas na execução e durante à depuração o estudante pode buscar informações sobre conceitos de determinada área.

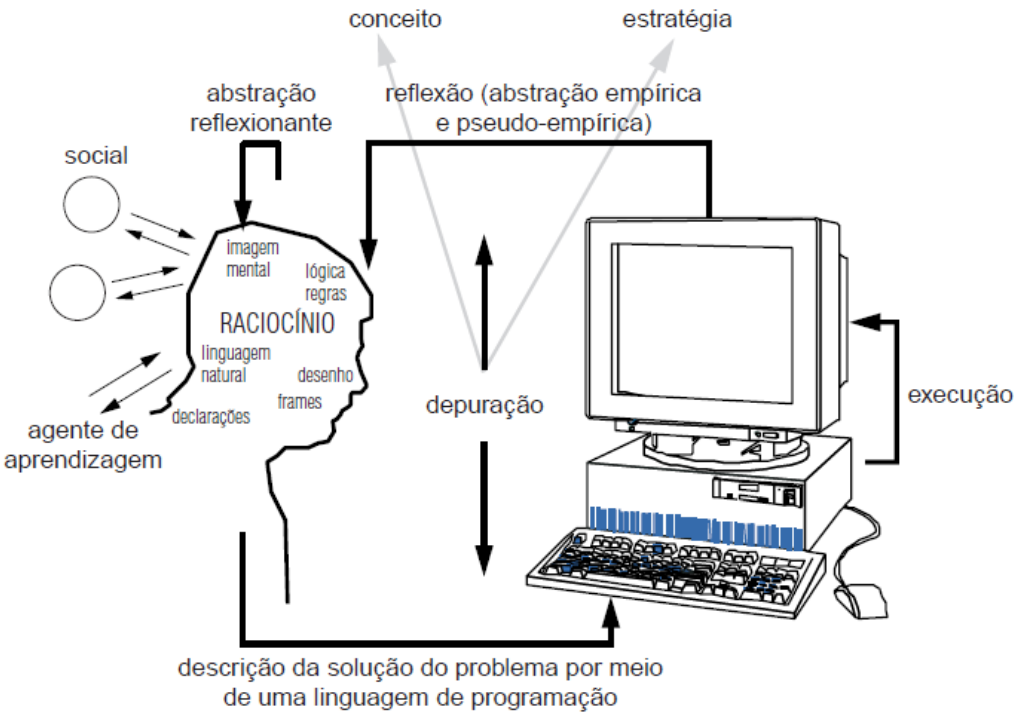


Figura 3-1: Interacção estudante-computador na situação de programação

(Fonte Valente 1999)

A fim de identificar um modelo para desenvolver ferramentas CASE educacionais, foi utilizada a plataforma de Jackson (1995) para análise de requisitos. De acordo com Emmerich, Finkelstein e Montangero (1997), a plataforma de Jackson nos permite distinguir entre os requisitos do domínio da aplicação, domínio da máquina e especificação, sendo que:

- **Domínio da aplicação** – São todas as características dos processos existentes no mundo real. Estas características estão divididas em dois grupos;
    - **Propriedades do domínio** – Conjunto de tudo que é verdade no domínio da aplicação independentemente do desenvolvimento do sistema desejado;
    - **Requisitos** – Conjunto de tudo que existe no domínio da aplicação, que se pretende tornar verdadeiro com o desenvolvimento do sistema desejado;
- **Domínio da máquina** – São todas as características pertencentes ao ambiente centrado aos processos que implementam o suporte solicitado;
- **Especificação** – São todas as características que identificam os fenómenos comuns entre os requisitos e os programas. São descrição do comportamento que o programa deve ter para satisfazer os requisitos;

Figura 3-2: Plataforma de Jackson para análise de requisitos

3.1.1 Propriedades do domínio

Para as instituições de ensino superior de Moçambique foram detectadas as seguintes propriedades de domínio:

- As instituições de ensino superior que leccionam cursos relacionados à ciência da computação em Moçambique têm no mínimo um laboratório de informática com computadores, que estão disponíveis às aulas de laboratório;
- No processo de cada disciplina de ensino e aprendizagem, as normas/notações utilizadas na análise, desenho, modelagem, integração, teste, manutenção ou qualquer outra actividade de desenvolvimento de *Software*, variam de instituição para instituição;

- Nem todos os computadores nos laboratórios têm o mesmo sistema operativo, os computadores do laboratório podem ter sistemas operativos diferentes dependendo das políticas da instituição;
- Nem em todas as instituições os estudantes apresentam uma conta privada nos computadores do laboratório;

3.1.2 Requisitos das ferramentas CASE educacionais

Devido ao facto da ferramenta CASE educacional ser uma ferramenta CASE e uma ferramenta educacional em simultâneo, ela deve cobrir aspectos técnicos e pedagógicos. Assim sendo, vários são os requisitos propostos para as ferramentas educacionais. No presente trabalho, a fim de propor os requisitos das ferramentas CASE foram abordados os requisitos propostos por Vannucci e Colla (2010), Thomas e Nejmeh (1992) e Wilson Filho (2001).

3.1.2.1 Requisitos de Vannucci e Colla

Vannucci e Colla (2010) identificaram seis requisitos que dizem respeito a aspectos técnicos e pedagógicos do *Software* educacional, os requisitos são os seguintes:

- **Interface eficiente** – A interface do produto e da interacção com o usuário deve ser clara e intuitiva.
- **Conteúdos multimédia de qualidade** – A qualidade tanto em termos de valor pedagógico e qualidade audiovisual são muito importantes, assim como ele pode determinar o grau de atenção que o utilizador presta aos conteúdos propostos.
- **Conteúdos apropriados** – os conteúdos propostos devem ser correctos e apresentados em uma forma adequada pelo respeito ao público a que são dirigidos. O *Software*, além disso deve ser capaz de adaptar seu conteúdo para os usuários, dirigindo-se a públicos de diferentes idades, objectivos e estilos de aprendizagem (van Dam 2005).
- **Abordagem multidisciplinar e extensões de conteúdos** – a natureza multidisciplinar e a presença de extensões de conteúdo é uma característica positiva para *Software* educacional à medida que se quer torná-lo capaz de atrair mais o interesse dos utilizadores. Além disso, as extensões de conteúdo permitem que os utilizadores escolham os temas que quer se aprofundar.

- **Modularidade** - modularidade permite os desenvolvedores facilmente mudar ou ampliar o conteúdo do *Software* produzido e aumentar desta forma a sua vida, além disso, se o *Software* inclui ferramentas de autoria, permite os professores transportar facilmente as mudanças para ajustar o produto às necessidades do usuário.
- **Portabilidade** – As ferramentas educacionais devem ser utilizadas em diferentes tipos de computadores, independentemente dos sistemas operativos e suas arquitecturas, dado que elas serão usadas tanto na instituição de ensino como em casa.

3.1.2.2 Requisitos de Thomas e Nejmeh

Thomas e Nejmeh (1992) consideram ser quatro as dimensões da integração de ferramentas CASE que devem ser analisadas: integração de dados, integração de controlo, integração de apresentação e integração de processos.

3.1.2.3 Requisitos de Wilson Filho

Wilson Filho (2001) propõe um conjunto de requisitos para os processos de desenvolvimento de *Software* educacional:

- **Arquitectura**: Um processo educacional deve ser composto por etapas, com critérios de entrada, entradas, actividades, saídas e os critérios de saída totalmente definidos;
- **Orientação a equipas:** um processo educacional deve ser projectado de forma que possa ser desenvolvido por pequenas equipas de estudantes (três a cinco estudantes);
- **Tempo de ciclo de projecto:** Um processo educacional deve apoiar todas etapas do ciclo de vida do projecto de *Software*;
- **Normas e práticas:** Um processo educacional deve expôr o aluno a normas, práticas e paradigmas extensamente reconhecidos;
- **Apoio ao estudante**: as normas exigidas devem ser fornecidas como parte do material do processo;
- **Apoio do Instrutor**: Um processo educacional deve conter elementos para oferecer suporte adequado para o trabalho de instrutor;

3.1.2.4 Requisitos propostos

Além dos requisitos para o desenvolvimento de *Software*, a ferramenta CASE educacional também deve suportar os requisitos das partes interessadas, deve satisfazer os requisitos do professor, dos alunos, da instituição e do país.

A fim de cumprir algumas das exigências propostas por Vannucci e Colla (2010) , Thomas e Nejmeh (1992) e Wilson Filho (2001), uma ferramenta CASE educacional deve ter os seguintes requisitos:

- **Ampla participação** – A ferramenta deve ser capaz de participar em todas ou grande parte das actividades da área para que foi projectado;
- **Fornecer tutoriais** – A ferramenta deve ter tutoriais que explicam como utilizar a mesma e apresentem alguns exemplos práticos;
- **Apoio à internacionalização** – A ferramenta deve apresentar um mecanismo simples para adição e remoção de línguas.
- **Seguir padrões** – A ferramenta deve seguir de forma fiel os padrões que utiliza;
- **Interagir com o utilizador** – A ferramenta deve interagir com o utilizador, através de mensagens de *feedback* (aviso, mensagens de erro e informação) ou outros meios.

3.1.3 Especificações da ferramenta CASE educacional

Nas instituições de ensino superior de Moçambique, os programas dos cursos têm um conjunto de cadeiras e cada cadeira está relacionada a alguns temas específicos do curso. Baseando-se nisso, uma ferramenta CASE educação ideal deve ser uma ferramenta CASE *Workbench*, que irá proporcionar uma boa experiência em desenvolvimento de *Software* durante o processo de ensino-aprendizagem na sala de aula.

As *Workbenches* integram em uma única aplicação, várias ferramentas de apoio as actividades específicas do processo de *Software* (Fuggett 1993). Assim, eles oferecem:

- Uma interface consistente e homogénea (integração de apresentação);
- Fácil invocação de ferramentas e cadeias de ferramentas de integração (controle);

- O acesso a um conjunto comum de dados, gerida de forma centralizada (integração de dados);

No entanto, ferramentas CASE *Workbenches* são divididas em oito classes, sendo cada classe relacionada com uma parte específica do processo de desenvolvimento de *Software.* As disciplinas relacionadas com o desenvolvimento de *Software* são diferentes de acordo com a sua abordagem, sendo que, cada disciplina apresenta os seus próprios requisitos que podem ser ajustados a uma das classes do *Workbench*.

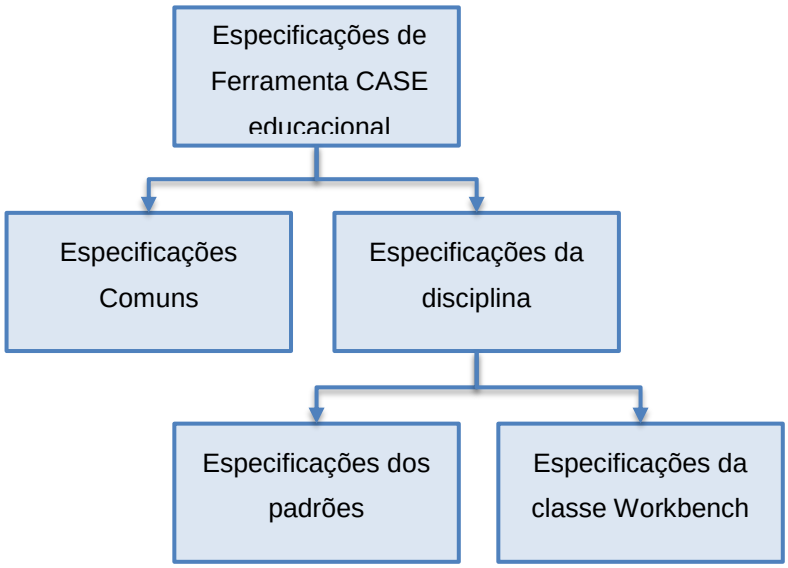


Figura 3-3: Especificações de ferramentas CASE educacionais

Neste ponto de vista, as especificações para a ferramenta CASE educação ideal estão divididos em dois grupos (Figura 3-3):

- **As especificações comuns -** são o grupo de especificação que todas as ferramentas CASE educacionais ideais devem ter, independentemente da disciplina para que foram projectadas. São especificações comuns as seguintes:
  - Permitir alterar as configurações;
  - Fornecer uma interface de utilizador amigável;
  - Exigir pouco conhecimento básico;
  - Tornar as disciplinas fáceis de aprender.
- **Especificações da disciplina**
  - **Especificações da classe de *Workbench* -** Estas especificações estão relacionadas com o assunto que a classe suporta;

- **Especificações dos padrões** – São especificações técnicas dos padrões utilizados na área de desenvolvimento suportada pela ferramenta;

3.2 Metodologia de desenvolvimento de ferramentas CASE educacionais

O desenvolvimento de *Software* para fins educacionais pode ser considerado mais complexo do que o desenvolvimento de *Software* corporativo. A fim de desenvolver uma ferramenta CASE educacional que vai ter sucesso nas disciplinas para que foi projectada, há muitos aspectos a ter em consideração:

- Os professores devem estar familiarizados com a ferramenta e reconhecê-la como uma ferramenta educacional. De acordo com Teixeira e Brandão (2003), *Software* educativo só faz sentido quando o professor reconhecê-lo como uma ferramenta capaz de auxiliá-lo no ensino, como ferramenta para o planeamento e realização de projectos interdisciplinares, como elemento que o motiva e o apoia nos desafios durante o desenvolvimento de novas práticas de ensino, fazendo com que a actividade de ensino-aprendizagem seja inovadora, dinâmica, participativa e interactiva.
- Deve-se utilizar técnicas para optimização da interface do utilizador durante o desenvolvimento da ferramenta, a fim de disponibilizar aos estudantes uma ferramenta simples de manipular.
- Os objectivos institucionais com o curso. Cada instituição tem alguns objectivos com os seus cursos, e esses objectivos podem mudar de instituição para instituição. Isso pode afectar os padrões utilizados para leccionar uma determinada disciplina, assim como os padrões podem ser diferentes em diferentes instituições.
- A análise da qualidade do *Software:* para *Software* educacional há factores inerentes ao contexto educacional, como questões culturais, éticas, filosóficas e psicopedagógicas, que influenciam a avaliação (Campos, Campos e Rocha 2007).

Considerando esses aspectos, uma boa metodologia para desenvolver ferramentas CASE educacionais não se deve concentrar apenas nos requisitos técnicos da ferramenta, deve interagir com todas as partes interessadas durante o processo de desenvolvimento e ter um mecanismo eficiente para validar os requisitos da ferramenta, antes de o processo de desenvolvimento terminar. Propõe-se uma

metodologia de desenvolvimento de *Software* educacional, baseada nas metodologias de prototipagem e de desenvolvimento incremental, onde a fase de prototipagem em cada incremento não inicia ates de o utilizador aprovar o protótipo da interface do utilizador desenhada.

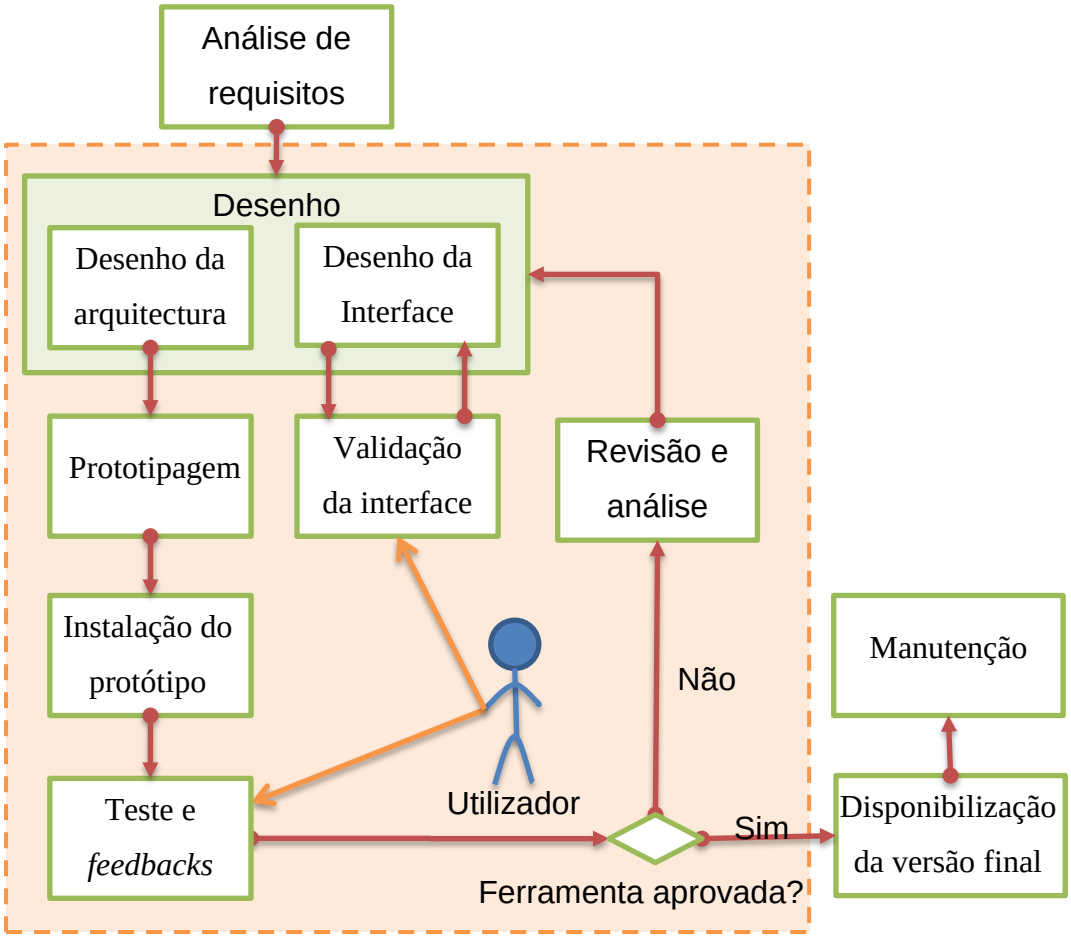


Figura 3-4: Metodologia de desenvolvimento de ferramentas CASE educacionais

A metodologia é composta pelas seguintes fases:

- **Fase de análise de requisitos**: Nesta fase faz-se o levantamento e análise dos requisitos da ferramenta, fazendo-se portanto o levantamento e análise dos requisitos técnicos e sociais;
- **Fase de desenho**: Nesta fase é feito o desenho da arquitectura do sistema e a interface do utilizador com base nos requisitos obtidos. Após desenhar a interface do utilizador, esta é sempre validada pelos utilizadores do sistema e redesenhada até que se obtenha uma aprovação dos mesmos. A fim de obter uma maior independência entre a interface do utilizador e a lógica de funcionamento da ferramenta, a arquitectura desenhada deve apresentar um baixo acoplamento (menor dependência) entre a parte da ferramenta

responsável pela interface do utilizador e a parte responsável pela lógica de funcionamento;

- **Fase de validação da interface**: Esta fase consiste em validar a interface do utilizador desenhada com um ou vários utilizadores (professores e alunos) antes do desenvolvimento do protótipo. A interface do utilizador pode ser validada somente com professores mas nunca somente com alunos, pois os alunos ainda não estão cientes da matéria a ser suportada pela ferramenta;
- **Fase de prototipagem**: Nesta fase é desenvolvido o primeiro protótipo funcional da ferramenta, baseando-se na arquitectura e interface do utilizador desenhada;
- **Fase de instalação do protótipo**: Nesta fase é disponibilizada a versão do protótipo desenvolvido de modo que os utilizadores interajam com o mesmo;
- **Fase de teste e *feedbacks***: Nesta fase os utilizadores interagem com o protótipo e apresentam *feedbacks* sobre a experiência que obtiveram com o protótipo e sobre o protótipo no geral.
- **Fase de revisão e análise**: Nesta fase é feita a análise do retorno obtido com os utilizadores e elabora-se a lista de requisitos a serem adicionados no próximo desenho do protótipo;
- **Fase de disponibilização da versão final**: Nesta fase é disponibilizada a versão final da ferramenta com toda a sua documentação;
- **Fase de manutenção**: Faz se a manutenção da ferramenta.

3.3 Senários para utilização de ferramentas CASE educacionais

Para uma boa utilização de ferramentas CASE educacionais em Moçambique, podem ser considerados os seguintes cenários:

- **Suporte ao professor**: Nesta abordagem, o professor funciona como um guia, onde ele fornece as informações necessárias sobre o tema que está leccionando e sobre como utilizar o suporte da ferramenta CASE, depois, o professor ajuda os alunos, durante o desenvolvimento de alguns projectos ou resolvendo alguns problemas;
- **Desenvolvimento de projecto**: Projectos da disciplina podem ser elaborados para apresentar as questões da vida real para os estudante e para transmitir níveis básicos de proficiência (W. P. Filho 2001). Nesta abordagem, os estudantes organizados em grupos ou individualmente, desenvolvem projectos

baseados na vida real ou em cenários hipotéticos, com ajuda da ferramenta CASE e do professor. A ferramenta dá-lhes mais dinâmica durante o processo de desenvolvimento, fornecendo recursos úteis e resultados mais fáceis de entender pelos colegas e pelo professor.

- **Auto-aprendizagem** - Neste cenário, o aluno baseando-se nas lições que teve na sala de aula, resolve alguns problemas relacionados com as aulas, com o apoio da ferramenta CASE para desenhar diagramas, codificação, análise de requisitos ou outra actividade de desenvolvimento de *Software*.

Capitulo IV – Caso de Estudo: Modelador de Base de Dados Relacional

O protótipo desenvolvido durante a realização do trabalho foi o Modelador de Base de Dados Relacional (RDBM, do inglês *Relational Database Modeler*), sendo esta uma ferramenta CASE para auxiliar o processo de ensino-aprendizagem nas aulas de projecto de base de dados relacionais na faculdade de engenharia da UEM. O RDBM destaca-se das ferramentas existentes por estar virado à aprendizagem, apresentado para tal vários mecanismos que possibilitam ao utilizador analisar os diferentes diagramas que modela.

4.1 Modelagem de base de dados na Faculdade de Engenharia da UEM

Na faculdade de engenharia da UEM as aulas de Projecto de Base de Dados (PBD) são leccionadas nas cadeiras de base de dados I e base de dados II, de forma a dotar os estudantes de conhecimentos sólidos sobre todas as fases do PBD (Figura 4-1).

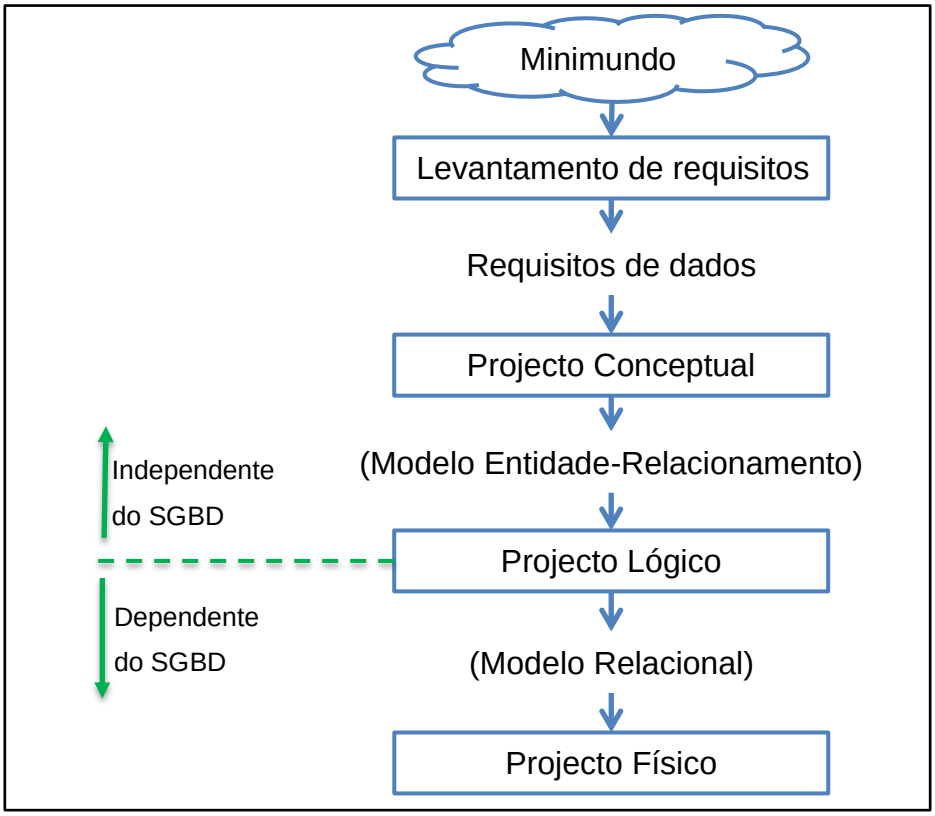


Figura 4-1: Fases do projecto de base de dados relacionais

Durante as aulas práticas e laboratoriais os estudantes entram em contacto com algumas ferramentas, como é o caso do *MySQL Workbench* (para o modelo relacional) na fase de projecto lógico e o *WampServer* na fase de projecto físico.

Apesar da utilização de algumas ferramentas durante as aulas, nenhuma ferramenta é utilizada para a fase de levantamento de requisitos e devido à não existência até o

momento de ferramentas que obedeçam de forma fiel à notação proposta por Elmasri e Navate (2003) (vide Anexo A2), notação essa utilizada para leccionar a modelagem de diagramas entidade relacionamento na Faculdade de Engenharia da UEM, vários são os meios alternativos utilizados na fase de projecto conceptual.

Alguns estudantes recorrem a folhas de papel ou ferramentas que possibilitam desenhar, como o *Microsoft Word*, *Paint* (programa padrão para desenhar, oferecidos pelo sistema operativo da *Microsoft*), o que lhes consome muito tempo e limita na concepção de bases de dados complexas. Outros estudantes recorrem a algumas ferramentas de modelagem existentes como o *brModelo* e do *BDDisigner,* o que requer que os estudantes dediquem um tempo extra para conhecer as notações utilizadas pelas outras ferramentas e as suas representações equivalentes com as das aulas.

4.2 Fase de análise de requisitos do RDBM

O RDBM visa auxiliar grande parte das actividades do PBD durante as aulas de base de dados, permitindo criar diagramas entidade-relacionamento e modelos lógicos e apresentar um mecanismo de análise e avaliação das actividades realizadas pelo utilizador.

O RDBM também apresenta um mecanismo automático para passagem de uma fase a outra, permitindo converter diagramas entidade-relacionamento em modelos relacionais e modelos relacionais em código SQL de um dado sistema de gestão de base de dados (SGBD). Permite também a engenharia inversa, saindo da fase de projecto físico à fase conceptual.

4.2.1 Requisitos funcionais do RDBM

Os requisitos funcionais do RBDM estão divididos em dois grupos, funções básicas e funções avançadas. As funções básicas são todas as funções consideradas fundamentais para que a ferramenta possa ser utilizada durante a fase de desenvolvimento e as funções avançadas sãos todas as funções dispensáveis nas primeiras versões da ferramenta.

A. Funções Básicas
   RF-A.1. Possibilitar abrir diagramas;
   RF-A.2. Possibilitar criar diagramas do nível conceptual;

RF-A.3. Possibilitar criar diagramas entidade-relacionamento na notação proposta por Elmasri e Navathe (2003);

RF-A.4. Possibilitar criar diagramas do nível lógico;

RF-A.5. Possibilitar adicionar componentes aos diagramas;

RF-A.6. Possibilitar remover componentes dos diagramas;

RF-A.7. Possibilitar editar as propriedades das componentes dos diagramas;

RF-A.8. Possibilitar gravar diagramas modelados;

RF-A.9. Emitir mensagens de alerta sempre que alguma regra de modelação for violada;

RF-A.10. Disponibilizar ficheiros de ajuda;

B. Funções avançadas

RF-B.1. Apresentar mecanismo automático para analisar a integridade dos diagramas modelados;

RF-B.2. Enviar registo de erros a equipe de desenvolvimento;

RF-B.3. Possibilitar a conversão de diagramas do nível conceptual ao nível lógico;

RF-B.4. Possibilitar a conversão de diagramas do nível lógico ao nível conceptual;

RF-B.5. Possibilitar alterar as propriedades da ferramenta;

RF-B.6. Possibilitar desfazer e refazer acções realizadas nos diagramas;

RF-B.7. Possibilitar gravar os diagramas em formato de imagem;

RF-B.8. Possibilitar imprimir diagramas;

RF-B.9. Possibilitar gerar ficheiros SQL através de diagramas do nível lógico;

RF-B.10. Possibilitar gerar diagramas do nível lógico através de ficheiros SQL;

RF-B.11. Possibilitar criar exercícios de projectos do nível conceptual;

RF-B.12. Possibilitar abrir exercícios criados;

RF-B.13. Possibilitar editar exercícios criados;

RF-B.14. Possibilitar gravar exercícios criados;

RF-B.15. Possibilitar modelar diagramas correspondentes a um dado exercício;

RF-B.16. Possibilitar analisar erros existentes nos diagramas correspondentes a dado exercício;

4.2.2 Requisitos não funcionais do RDBM

Os requisitos não funcionais geralmente se aplicam ao sistema como um todo, em vez de funções ou serviços individuais do sistema. Estas são restrições sobre os serviços ou funções oferecidos pelo sistema (Sommerville 2011).

A. Requisitos operacionais
   RNF-A.1. Deverá ser capaz de operar em ambiente Windows, Macintosh e Linux;
   RNF-A.2. Deverá ser capaz de ler e escrever em documentos XML;
   RNF-A.3. Deverá ser capaz de exportar arquivos gráficos GIF, JPG e BMP;
B. Requisitos de desempenho
   RNF-B.1. As operações que alteram as propriedades das componentes dos diagramas devem ser executas em tempo real, isto é, o resultado da sua execução deve ser disponibilizado imediatamente, no menor tempo possível;
   RNF-B.2. As componentes devem ser adicionadas ou removidas em tempo real;
C. Requisitos de padrões
   RNF-C.1. Deve permitir modelar diagramas no nível conceptual sem violar a notação proposta;
D. Requisitos de usabilidade
   RNF-D.1. Apresentar uma interface amigável;
   RNF-D.2. Requerer mínimo conhecimento prévio;
   RNF-D.3. Disponibilizar janelas de ajuda;

4.2.3 Casos de uso do RDBM

O diagrama de casos de uso apresentado baseia-se nos requisitos levantados para a ferramenta. Na Figura 4-2 estão representados os casos de uso das principais funções do MDBR (as suas descrições encontra-se no Anexo A3).

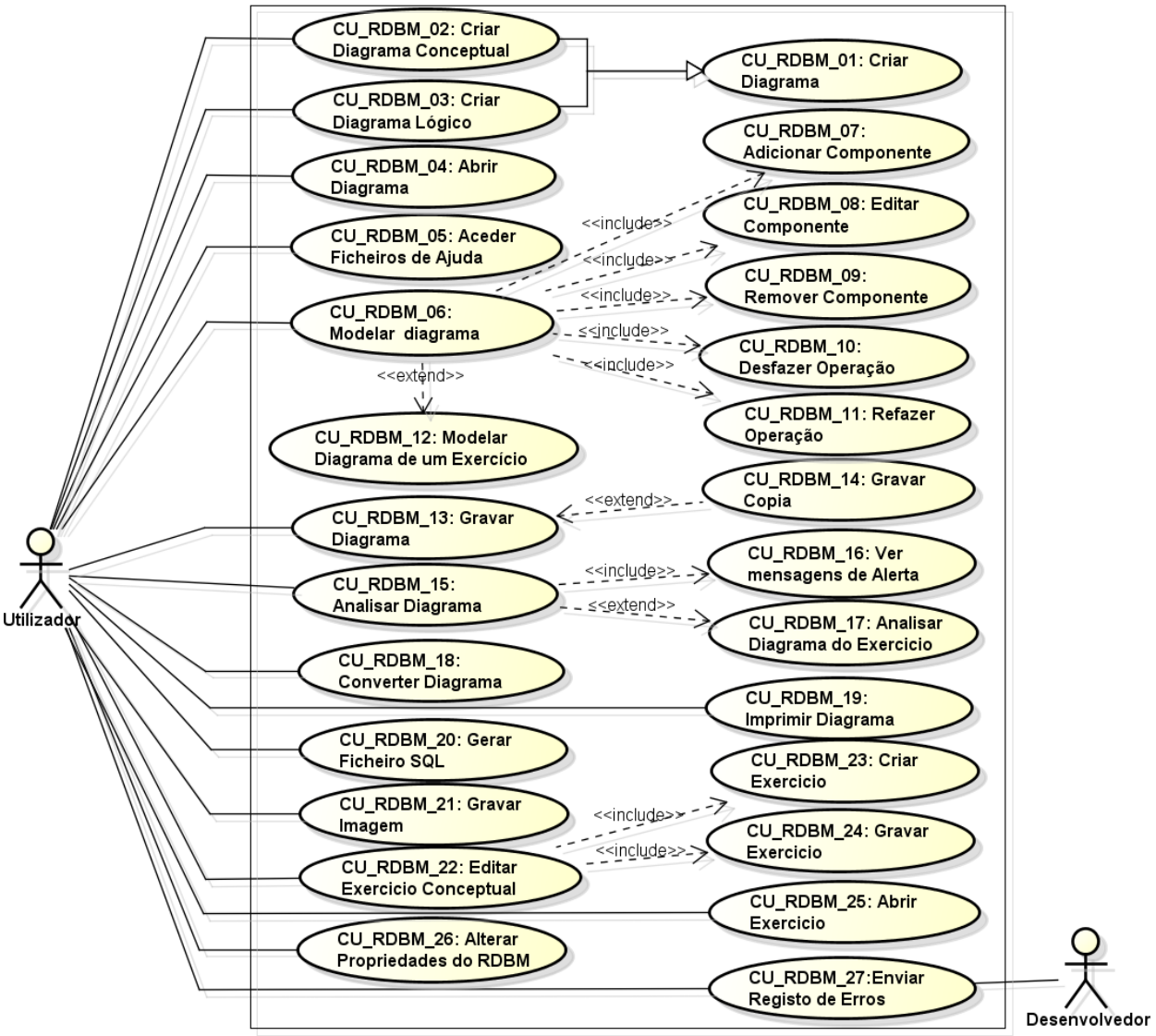


Figura 4-2: Modelador de Base de Dados Relacional - Diagrama de caso de uso

4.3 Fase de desenho do RDBM

4.3.1 Desenho da interface do utilizador

Durante a fase do desenho da interface do utilizador foram modeladas duas interfaces (vide Anexo A4) antes da obtenção da interface final.

A primeira interface foi inspirada na interface *do Socio-Technical Security Tool*[2], a segunda interface foi elaborada após algumas sugestões do professor Marco Ronchetti e a terceira interface (Figura 4-3) foi elaborada após a última validação efectuada com o professor Vali Issufo.

[2] Socio-Technical Security Tool é uma ferramenta para especificação de requisitos de segurança e confiabilidade de sistemas operando em um ambiente inter-organizacional

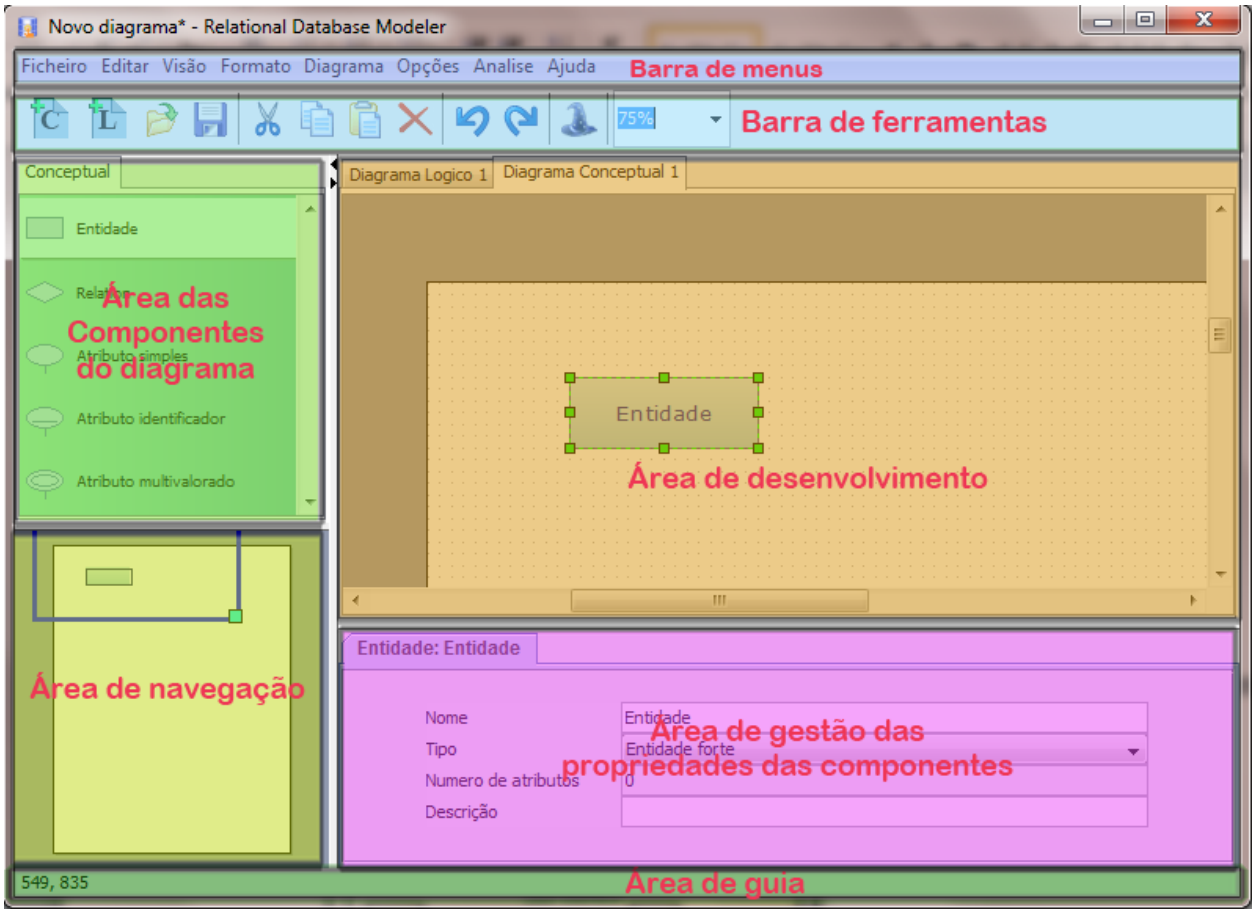


Figura 4-3: Modelador de Base de Dados Relacional - Interface principal

A interface principal do programa esta basicamente divida em sete áreas (Figura 4-3):

- **Área das componentes do diagrama** – Tem as componentes utilizadas para criarem diagramas de nível conceptual ou lógico, dependendo do tipo de diagrama que está sendo modelado.
- **Área de desenvolvimento –** esta área apresenta todos os projectos que estão sendo desenvolvidos no momento. É nesta área onde os diagramas são modelados.
- **Área de gestão das propriedades das componentes** – esta área possibilita alterar as propriedades das componentes do diagrama.
- **Área de guia** – esta área apresenta informações que possam guiar o utilizador, como por exemplo a posição do rato na área de desenvolvimento e descrição das actividades que a aplicação possa estar executando em *background*.
- **Área de navegação** – Tem as funções que possibilitam navegar pelo diagrama. Apresenta uma árvore com as componentes do diagrama e uma pequena visão do diagrama que possibilita ampliar o diagrama.
- **Barra de menus** – Contem os menus do programa.

- **Barra de ferramentas** – Contem operações básicas do programa (Tabela 4-1);

Ícone	Operação	Descrição
	Novo diagrama conceptual	Cria uma nova área de trabalho para modelar diagrama do nível lógico.
	Novo diagrama lógico	Cria uma nova área de trabalho para modelar diagrama do nível lógico.
	Abrir diagrama	Abri um projecto existente.
	Gravar diagrama	Grava o projecto.
	Cortar	Corta a componente seleccionada.
	Copiar	Copia as componentes seleccionadas.
	Colar	Cola as componentes copiadas.
	Apagar	Apaga as componentes seleccionadas.
	Desfazer	Desfaz uma alteração.
	Refazer	Refaz uma operação de desfazer.
	Converter	Converte um diagrama do nível conceptual ao um diagrama do nível lógico ou um diagrama do nível lógico a um diagrama do nível conceptual.

Tabela 4-1: Modelador de Base de Dados Relacional - Componentes da barra de ferramentas

4.3.2 Desenho da arquitectura

Uma arquitectura ideal para ferramentas CASE educacionais, é aquela que possibilita a separação de requisitos relacionados com ciências de computação dos requisitos sociais.

A arquitectura deste RDBM é uma arquitectura com três camadas: Camada de gestão de dados, camada de negócios e camada da interface do utilizador. As camadas inferiores apresentam uma interface de comunicação com a camada imediatamente superior, possibilitando assim que as camadas possam ser modificadas sem afectar o funcionamento da camada superior (caso a modificação não altere a interface de comunicação).

Esta estruturação em camadas tem como objectivo facilitar a alocação da funcionalidade aos componentes (A. M. Filho 2007). O uso de camadas oferece suporte à flexibilidade e portabilidade, o que resulta em facilidade de manutenção (A. M. Filho 2007). Esta organização das camadas fornece uma arquitectura que possibilita desenvolver toda a lógica persistente da aplicação sem precisar de pensar nos requisitos que irão alterar com o tempo, uma vez que a lógica para criar diagramas

entidade-relacionamento e modelar diagramas relacionais não irá se alterar durante o desenvolvimento.

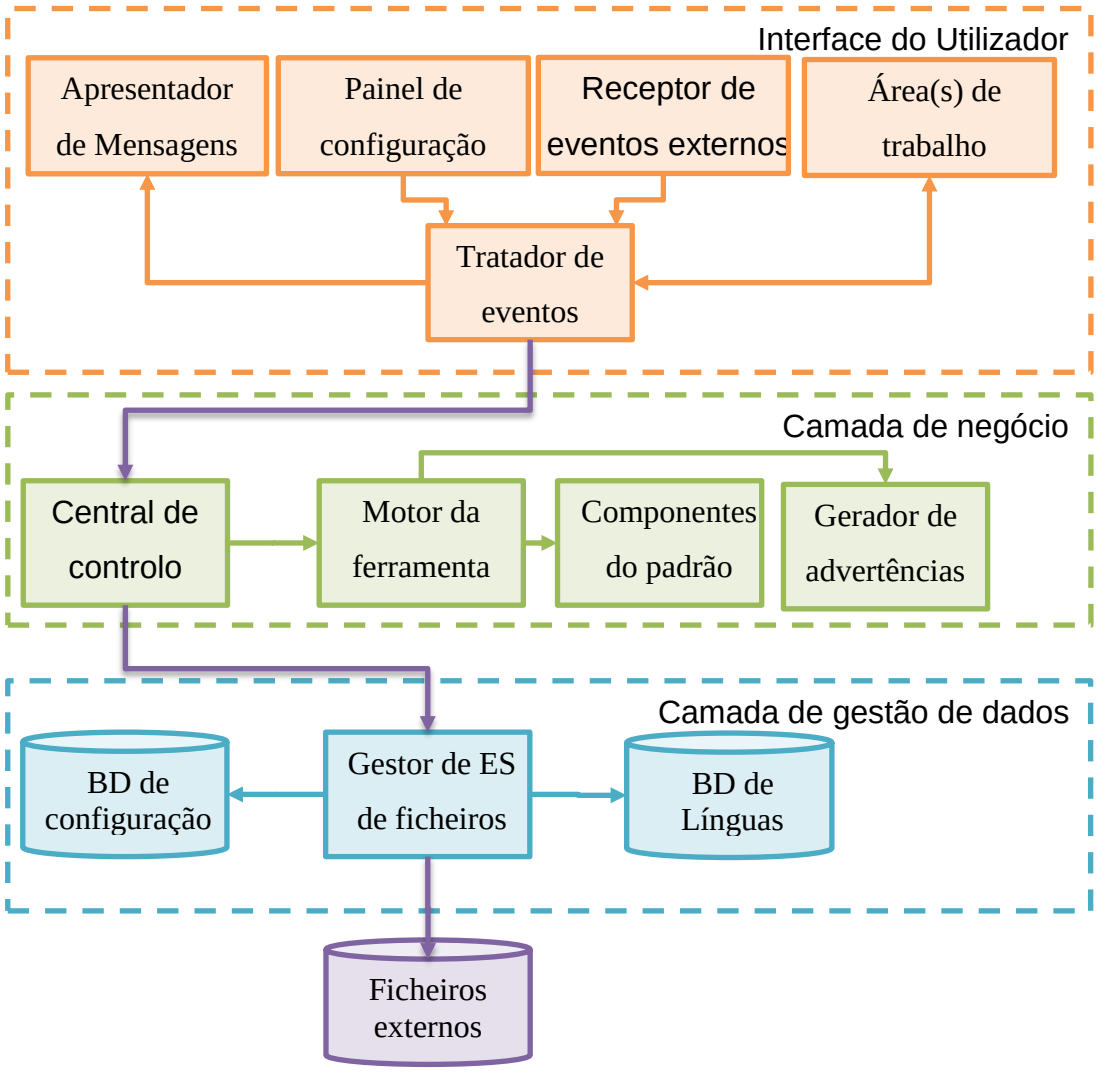


Figura 4-4: Arquitectura do RDBM

4.3.2.1 Camada de gestão de dados

A camada de gestão de dados é independente de todas as outras camadas, disponibilizando somente métodos que possibilitam gravar e aceder a ficheiros. O facto da camada de gestão de dados ser independente das outras camadas, possibilita que as outras camadas sejam alteradas sem á afectar.

4.3.2.2 Camada de negócio

A camada de negócio é a camada responsável por gerir a logica de funcionamento da ferramenta. É independente da camada da interface do utilizador, o que possibilita que ela seja desenvolvida independentemente de aspectos sociais relacionados a

usabilidade da ferramenta. De uma forma generalizada esta camada é composta por quatro componentes:

- **Gerador de advertências** – É responsável por gerar mensagens de respostas das solicitações do utilizador. As mensagens podem ser de sucesso, erro, aviso ou outras.
- **Componentes do padrão** – São elementos pertencentes ao padrão ou padrões que a ferramenta segue.
- **Motor da ferramenta** – É uma das componentes mais importantes da ferramenta, é responsável por realizar as actividades para as quais a ferramenta foi projectada, quer seja análise de requisitos, desenho de arquitecturas, programação, teste de componentes ou outras actividades do processo de desenvolvimento de *Software*.
- **Central de controlo** – É responsável por gerir todas as solicitações feitas pela camada da interface do utilizador. A central de controlo após receber alguma solicitação encaminha o pedido para a componente responsável pela actividade.

4.3.3 Camada da interface do utilizador

A camada da interface do utilizador é a camada que apresenta todos os elementos responsáveis pela interacção com o utilizador, esta camada interage com a camada de negócios a fim de responder as solicitações do utilizador. Todas as chamadas efectuadas da camada da interface do utilizador à camada de negócios são feitas através do tratador de eventos que se comunica com a central de controlo através dos métodos que esta tem disponíveis. Este mecanismo de comunicação possibilita que a camada de negócio possa beneficiar de actualizações sem afectar a camada da interface do utilizador desde que os parâmetros de entrada e de saída dos métodos disponibilizados não se alterem.

A camada de apresentação é composta pelos seguintes elementos:

- **Área(s) de trabalho** – As áreas de trabalho, são as componentes pertencentes a interface da ferramenta, as quais disponibilizam ao utilizador um ambiente para desenvolver as actividades para as quais a ferramenta está destinada;
- **Eventos externos** – É responsável por informar a ferramenta sobre qualquer evento externo que pode afectar o funcionamento da ferramenta.

- **Painel de configuração** – É a componente pela qual o utilizador pode alterar ou personalizar as configurações da ferramenta.
- **Apresentador de mensagens** – É responsável por apresentar as mensagens de resposta ao utilizador.
- **Tratador de eventos** – É responsável por responder a todas as ocorrências externas. É o tratador de eventos que recebe as solicitações do utilizador e as encaminha à camada de negócio.

4.3.4 Diagramas de classes do RDBM

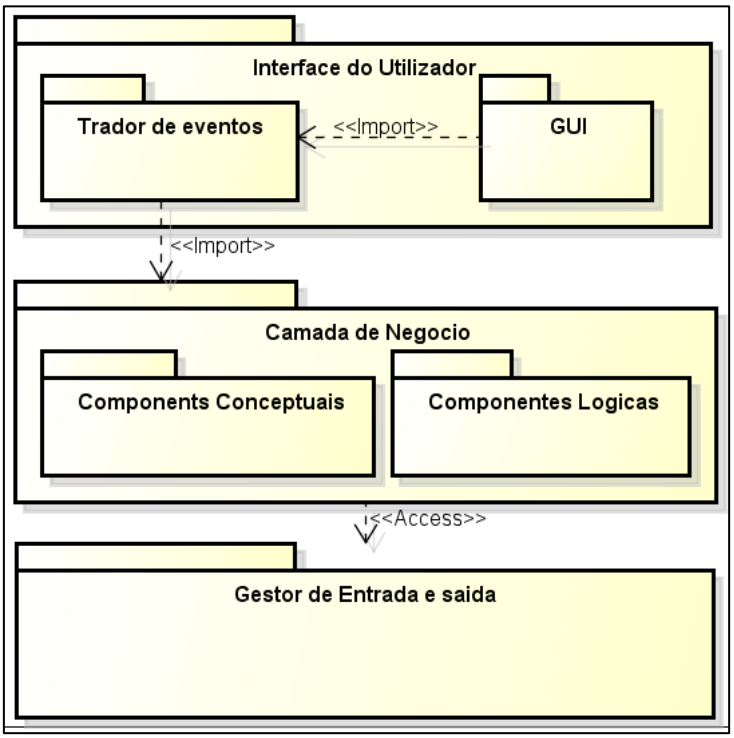


Figura 4-5: Modelador de Base de Dados Relacional - Vista de pacotes do diagrama de casses

4.3.4.1 Pacote de gestão das entradas e saídas

Este pacote é responsável por aceder os ficheiros de configuração do programa e possibilitar gravar e abrir projectos.

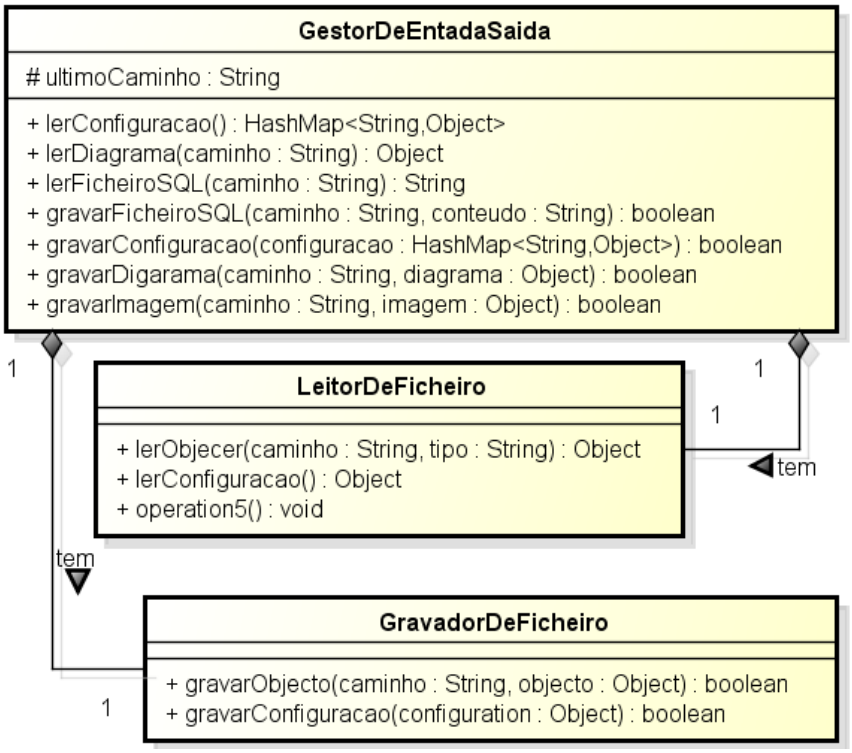


Figura 4-6: Modelador de Base de Dados Relacional - Pacote de entrada e saída

4.3.4.2 Pacote da camada de negócio

O pacote da camada de negócio é composto pelas classes que possibilitam a modelagem de diagramas, conversão de diagramas, geração de código SQL, geração de mensagens e outras operações relacionadas com a logica de funcionamento da ferramenta. Este pacote apresenta duas classes abstractas que são as classes elemento e contentor, a classe elemento representa cada componente que pertence ao diagrama e o contentor representa uma área abstracta na qual os diagramas são modelados. Além das duas classes abstractas, o pacote da camada de negócio contem um motor que é responsável por gerir as operações da camada de negócio.

O pacote de negócio apresenta dois outros pacotes, nomeadamente o pacote conceptual e o pacote lógico, sendo que cada um deles é responsável pela logica de funcionamento de um tipo de diagrama.

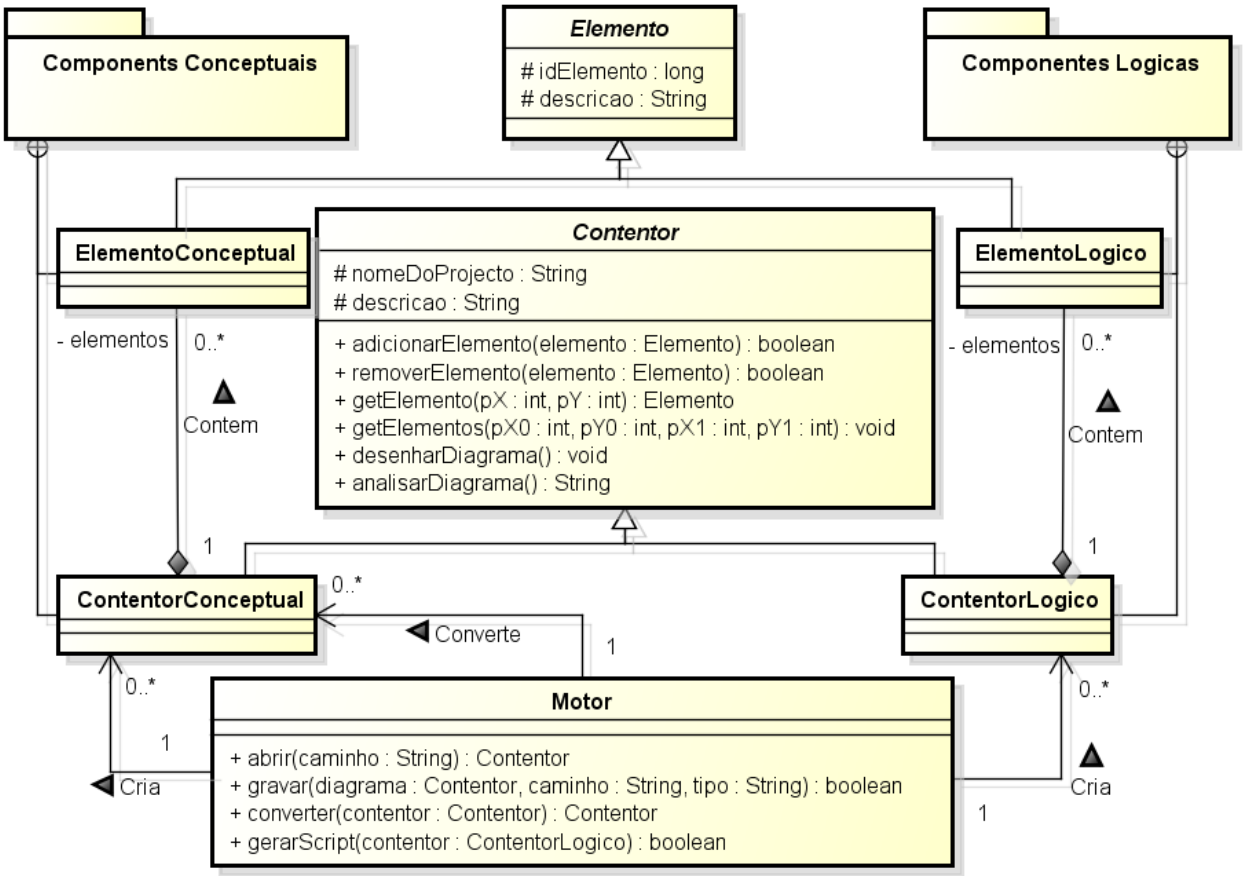


Figura 4-7: Modelador de Base de Dados Relacional - Pacote de negócio

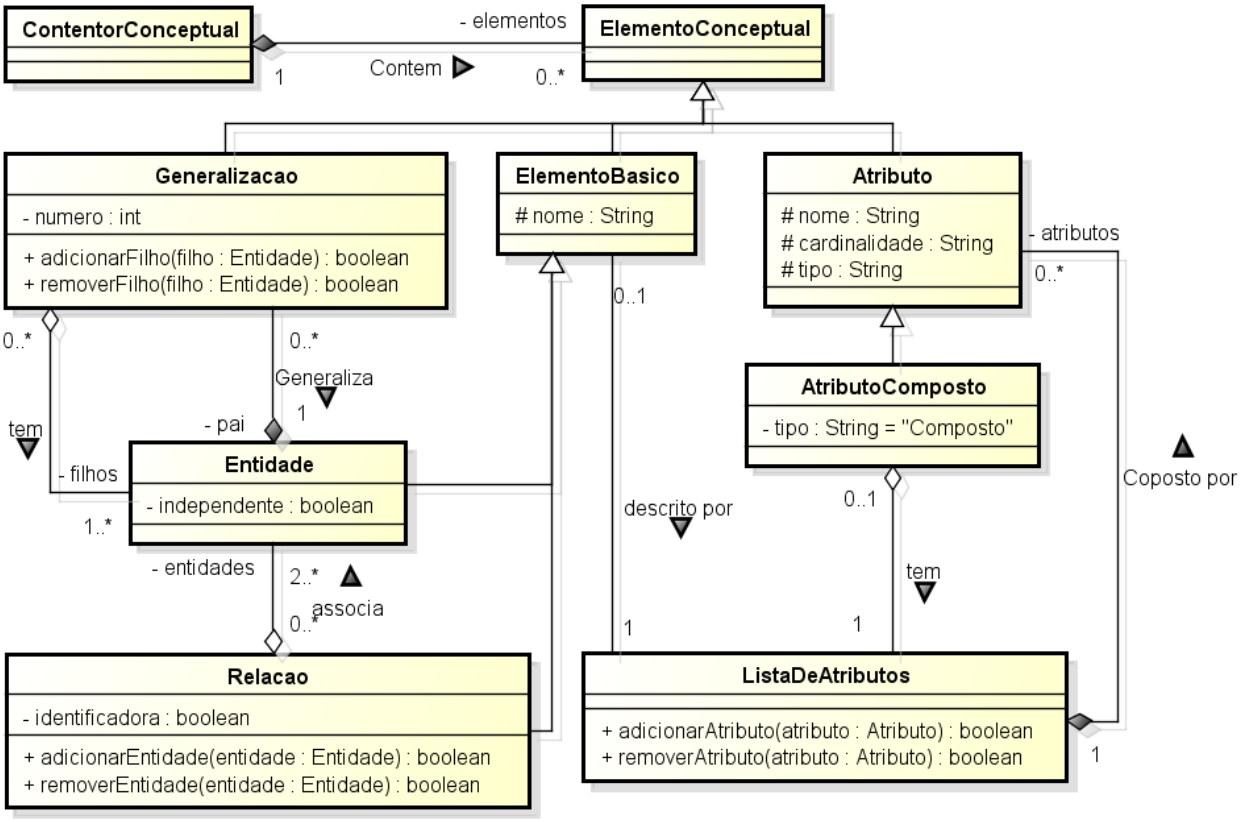


Figura 4-8: Modelador de Base de Dados Relacional - Pacote conceptual

O pacote conceptual e o pacote lógico contêm componentes que possibilitam a modelagem de diagramas dos níveis conceptual e lógico respectivamente. Cada um deles, além das componentes que possibilitam a modelagem, é composto por contentores que funcionam como controladores para monitorar e restringir operações não autorizadas durante o processo de modelagem. O pacote conceptual segue o meta-modelo para modelagem de diagramas entidade-relacionamento proposto por Atzeni et al (1999).

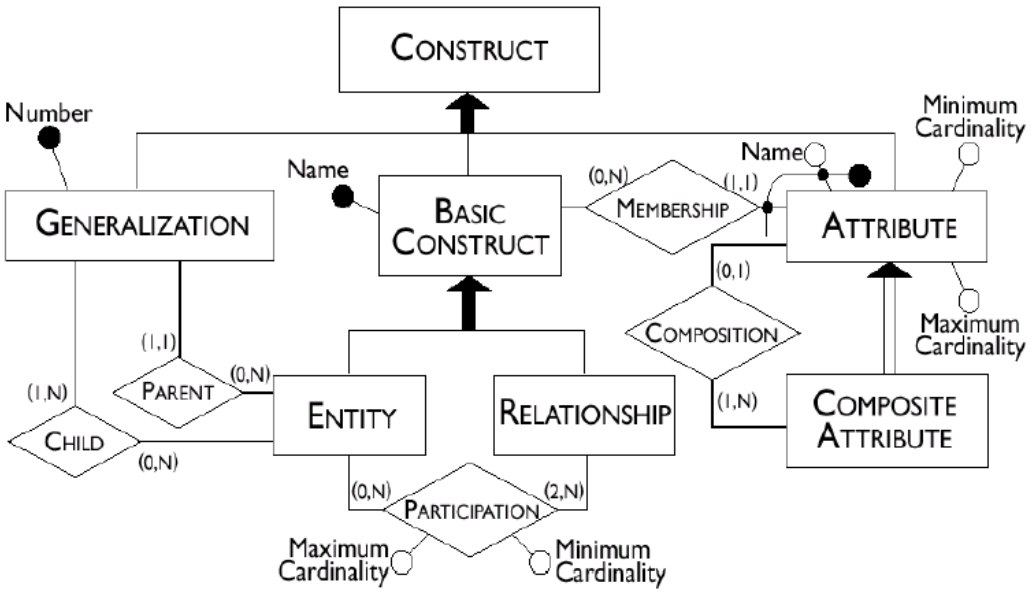


Figura 4-9: Meta-modelo para modelagem de diagramas ER apresentado (fonte: Atzeni et al, 1999)

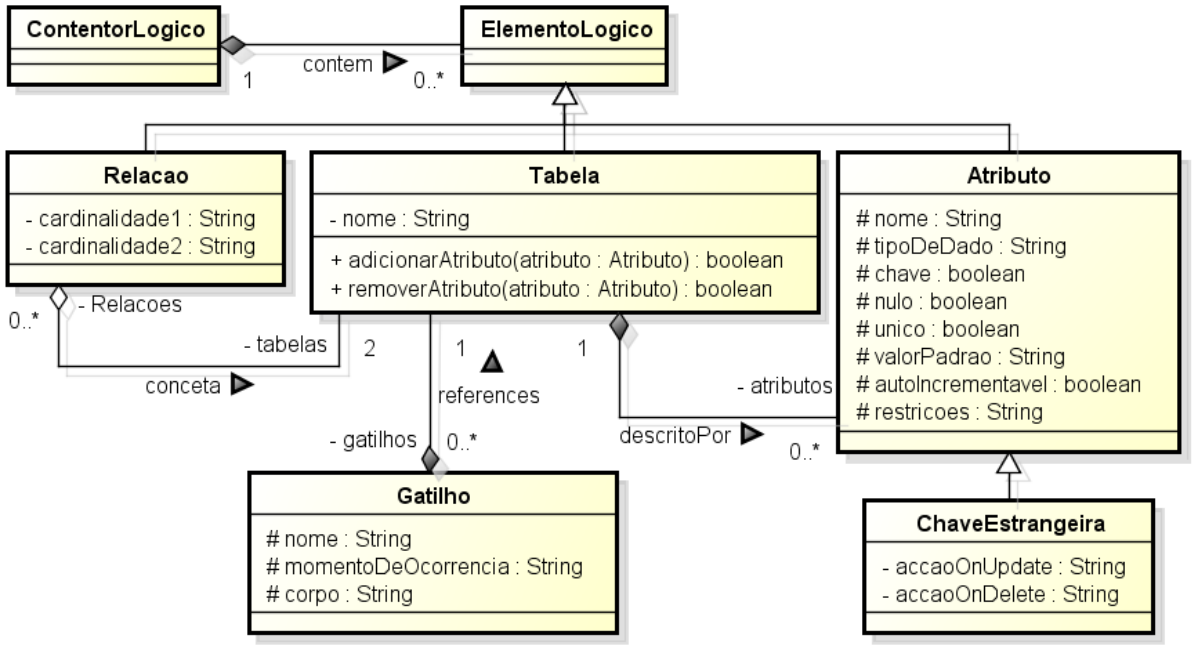


Figura 4-10: Modelador de Base de Dados Relacional - Pacote lógico

4.3.4.3 Pacote da interface do utilizador

O pacote da interface do utilizador contem todas as classes que são responsáveis pela interacção directa com o utilizador, essas são as classes relacionadas com a interface gráfica do utilizador (GUI) e os tratadores de eventos. Este pacote é composto por dois outros pacotes, pacote de tratamento de eventos e pacote da interface gráfica do utilizador.

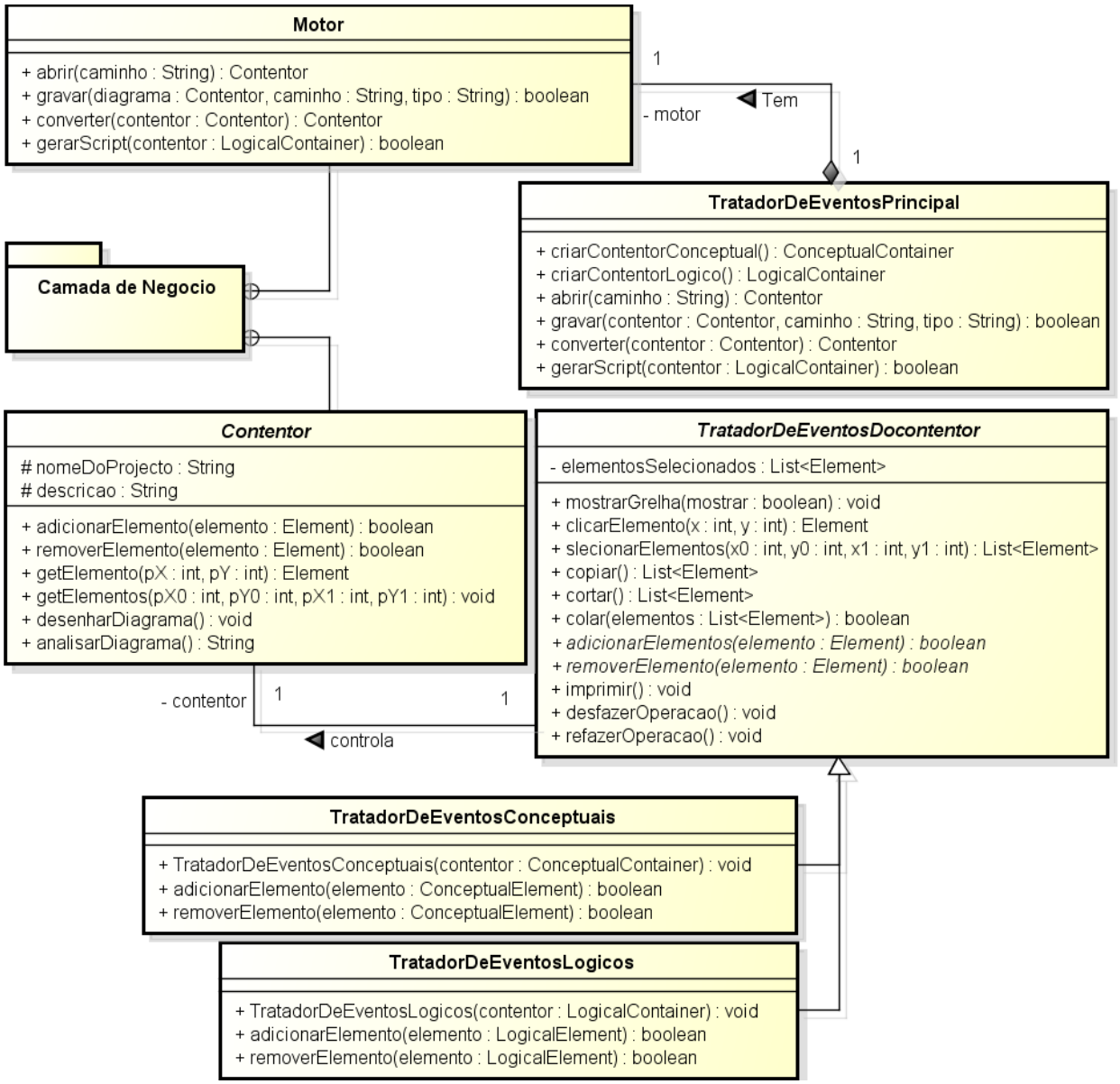


Figura 4-11: Modelador de Base de Dados Relacional - Pacote de tratamento de eventos

O pacote de tratamento de eventos (Figura 4-11) contem todas as classes responsáveis por tratar os eventos, o tratador de eventos principal é responsável por cuidar de todos os eventos relacionados com o programa no geral e o tratador de

eventos do contentor é responsável por cuidar de todos os eventos relacionados com o processo de modelagem.

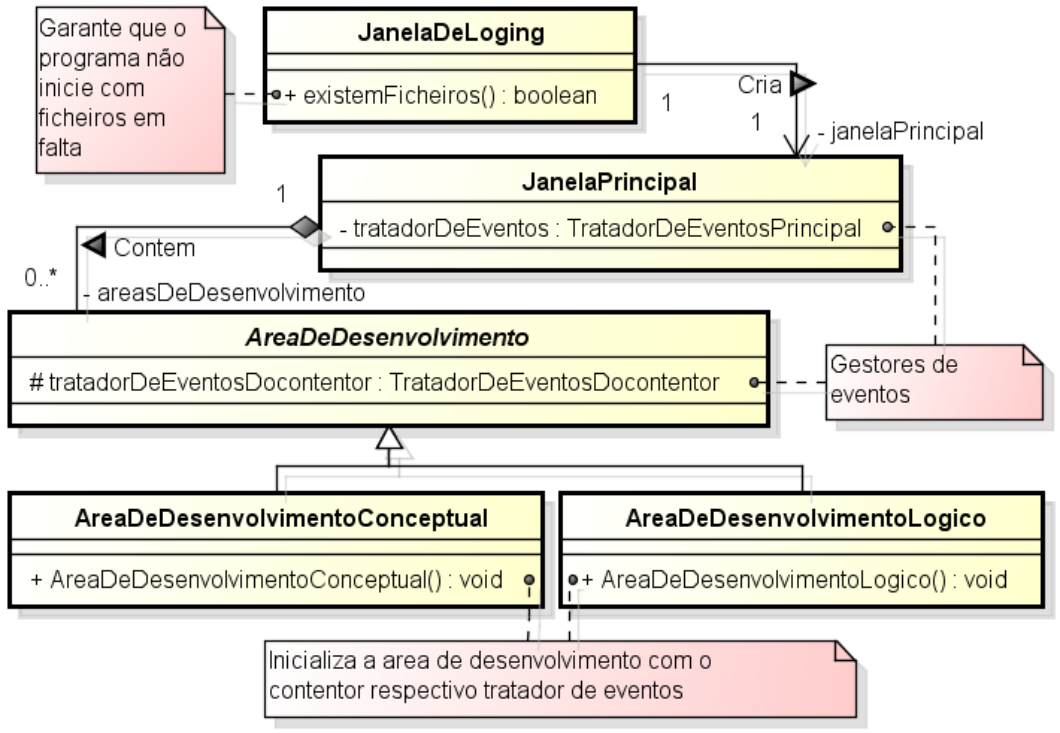


Figura 4-12: Modelador de Base de Dados Relacional – Pacote da interface gráfica do utilizador

O pacote da interface gráfica do utilizador (Figura 4-12) é constituído pelas classes responsáveis por disponibilizarem a interface gráfica ao utilizador, sendo que, para poderem responder as solicitações do utilizador, algumas dessas classes contêm tratadores de eventos.

4.4 Fase de prototipagem

A fim de obter-se uma ferramenta que satisfaça os requisitos reais mais relevantes para uma ferramenta que auxilia o processo de ensino-aprendizagem durante as aulas de PBD, para o primeiro protótipo foram seleccionados somente alguns dos requisitos da ferramenta, com o objectivo de garantir que as funcionalidades da ferramenta satisfaçam o seu propósito.

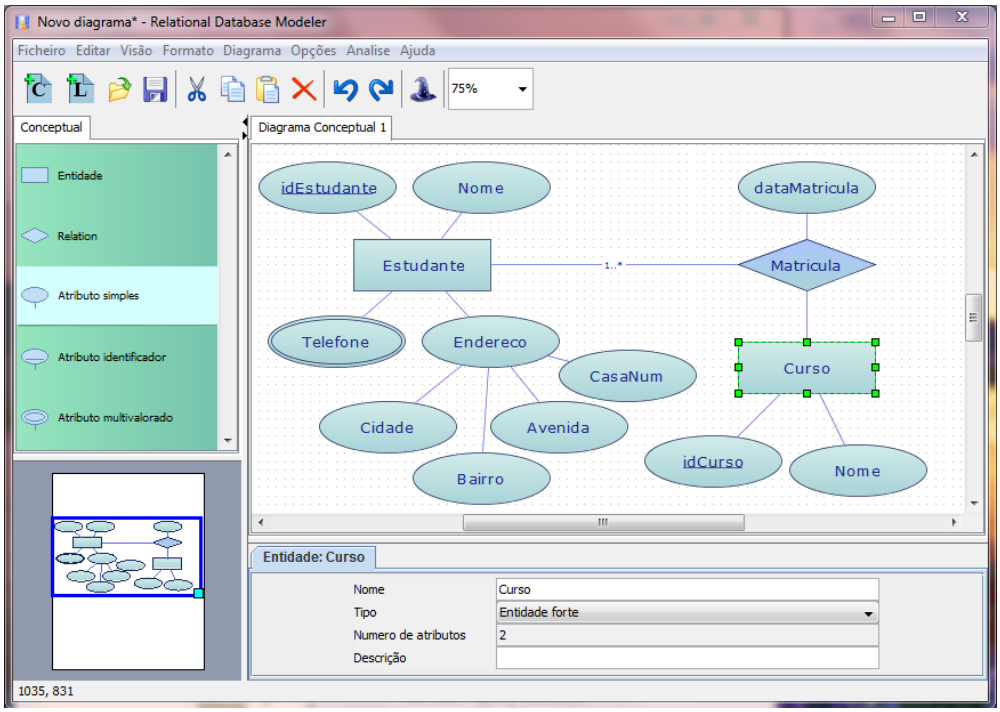


Figura 4-13: Primeiro protótipo do RDBM

4.4.1 Requisitos do primeiro protótipo do RDBM

O primeiro protótipo (vide Anexo A5) auxilia a fase do projecto conceptual, assim sendo os requisitos seleccionados para o seu desenvolvimento estão relacionados com a modelagem de diagramas entidade-relacionamento. Abaixo apresenta-se a lista dos requisitos do protótipo desenvolvido (o número de referencia dos requisitos do protótipo corresponde ao número de referencia dos requisitos da ferramenta).

A - Funções Básicas

RF-A.1 Possibilitar abrir diagramas;

RF-A.2 Possibilitar criar diagramas do nível conceptual;

RF-A.3 Possibilitar criar diagramas entidade-relacionamento na notação proposta por Elmasri e Navathe ;

RF-A.5 Possibilitar adicionar componentes aos diagramas;

RF-A.6 Possibilitar remover componentes dos diagramas;

RF-A.7 Possibilitar editar as propriedades das componentes dos diagramas;

RF-A.8 Possibilitar gravar diagramas modelados;

RF-A.9 Emitir mensagens de alerta sempre que alguma regra de modelação for violada;

B - Funções avançadas

RF-B.1 Apresentar mecanismo automático para analisar a integridade dos diagramas modelados;
RF-B.6 Possibilitar desfazer e refazer acções realizadas nos diagramas;
RF-B.7 Possibilitar gravar os diagramas em formato de imagem;
RF-B.8 Possibilitar imprimir diagramas;

4.4.2 Tecnologias utilizadas

O protótipo foi desenvolvido utilizado a linguagem Java por ser uma linguagem multiplataforma, possibilitando que o mesmo possa ser executado em sistemas operativos diferentes (requisito não funcional RNF-A.1).

Para o desenho da interface utilizou-se o compilador *Netbeans* por ser um IDE (ambiente de desenvolvimento integrado) que possibilita programar na linguagem Java e que o autor vem trabalhando a anos (na faculdade, no estagio e serviços particulares). O código seguiu os conceitos de programação orientada a objectos (POO), com o objectivo de desfrutar de todas as vantagens disponibilizadas pela POO, como reuso de código e fácil compreensão.

Para desenhar as componentes gráficas na tela, utilizou-se uma API (Interface de Programação de Aplicações). A API utilizada foi o JGraph, por ser uma API *open source*, dispor de comunidades para suporte aos desenvolvedores e ser rica em recursos gráficos.

4.4.3 Motor de análise

O RDBM destaca-se por ser uma ferramenta CASE direccionada para o processo de ensino-aprendizagem e para garantir que o RDBM tenha uma participação mais dinâmica nesse processo, foi embutido um motor de análise no mesmo. O motor de análise possibilita que durante o processo de modelagem a ferramenta possa advertir sobre, ou ilustrar ao estudante, os erros que foram cometidos, permitido assim que o utilizador possa reflectir sobre a estrutura do diagrama modelado e daí aperfeiçoá-lo.

Para o primeiro protótipo o motor de análise é capaz de detectar alguns erros básicos (Figura 4-14).

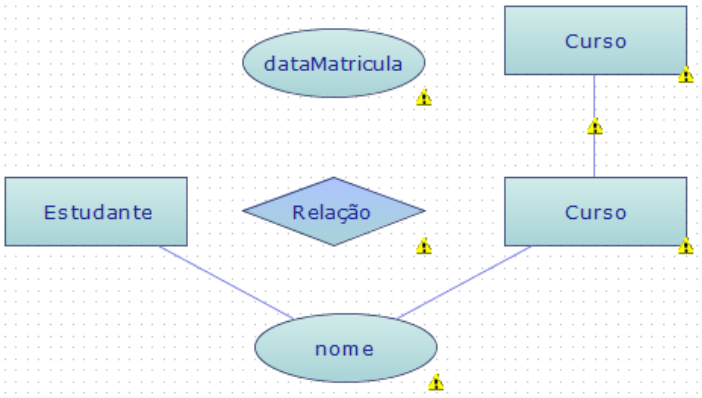


Figura 4-14: Diagrama ER utilizado durante os testes do RDBM

Os erros básicos detectados pelo motor de análise são:

1) Duplicação de nomes de elementos (Figura 4-16);
2) Conexão incorrecta entre componentes (Figura 4-15 e Figura 4-19);
3) Elementos soltos (Figura 4-17 e Figura 4-18);

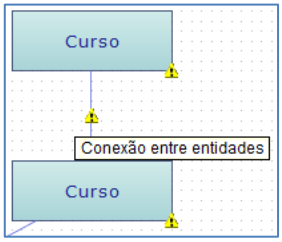


Figura 4-15: RDBM - Teste de Análise - Conexão entre entidades

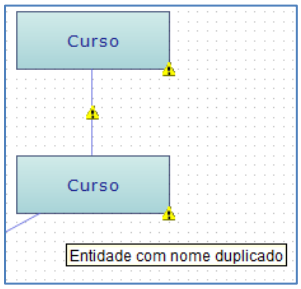


Figura 4-16: RDBM - Teste de Análise - Nome duplicado

Figura 4-17: RDBM - Teste de Análise - Relação solta

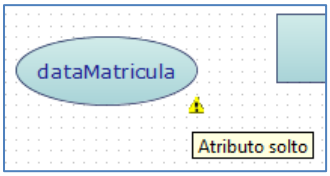


Figura 4-18: RDBM - Teste de Análise - Atributo solto

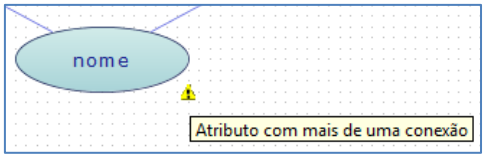


Figura 4-19: RDBM - Teste de Análise - Atributo com mais de uma conexão

Capitulo V – Discussão dos Resultados

No presente trabalho fez se o estudo sobre uma metodologia para desenvolvimento de ferramentas CASE educacionais em Moçambique. O trabalho teve como bases a revisão de literatura (capitulo II), análise de teorias (capitulo III) e implementação dos resultados (capitulo IV).

Na revisão de literatura abordou-se aspectos técnicos relacionados as ferramentas CASE e a participação das ferramentas CASE na educação. Durante esta fase, foi possível notar através de varias fontes que as ferramentas CASE podem ser divididas em vários subgrupos de acordo com as fases e actividades do desenvolvimento de *Software* que elas suportam e também foi possível notar que as ferramentas CASE são importantes no processo de ensino-aprendizagem.

Com a revisão de literatura, foi possível analisar os aspectos teóricos referentes ao desenvolvimento de ferramentas CASE educacionais. Com os resultados dessa análise, foi possível identificar os requisitos básicos das ferramentas CASE educacionais, propor uma metodologia para desenvolvimento das ferramentas CASE e identificar os possíveis cenários para utilização das ferramentas CASE educacionais em Moçambique.

Na fase de implementação, baseou-se nos resultados obtidos na análise das teorias para desenvolver um protótipo de uma ferramenta CASE educacional para auxiliar as aulas de PBD na cadeira de base de dados (o RDBM). Durante esta fase definiu-se os requisitos, os casos de uso, a interface do utilizador e a arquitectora do RDBM.

Para definir a interface do utilizador do RDBM, onde foram desenhadas três versões da interface do utilizador, baseou-se nos *feedbacks* obtidos com os professores Marco Ronchetti e Vali Issufo. A primeira interface desenhada baseou-se na interface do *Socio-Technical Security Tool*, sendo de notar que essa interface continha algumas funcionalidades desnecessárias para o processo de ensino-aprendizagem de modelação de base de dados relacionais (como é o caso de possibilitar que os alunos alteram o formato das figuras). A segunda interface do utilizador baseou-se na primeira (acima referida), retirando-se as funcionalidades desnecessárias. A ultima interface do utilizador, é similar à segunda interface, sendo que, conta com menus e funcionalidades adicionais.

A arquitectura apresentada para o RDBM, é uma arquitectura ideal para ser utilizada como base no desenvolvimento de ferramentas CASE educacionais para modelagem de diagramas, pois esta possibilita uma maior independência entre os diferentes níveis de desenvolvimento da ferramenta e permite integração futura de novas componentes.

O protótipo produzido auxilia a fase do projecto conceptual do PDB, apesar de não ter sido implementado em versão final, o protótipo desenvolvido constitui um resultado importante, tendo vantagens comparativas pois diferencia-se da maioria das ferramentas existentes por ter fins educacionais, apresentar requisitos de ferramentas educacionais e por ter um motor capaz de analisar em tempo real o diagrama que está sendo modelado e alertar ao estudante de alguns erros básicos que ele pode cometer durante a modelagem (por exemplo: entidades com nomes repetidos, etc.).

Capitulo VI – Conclusões e recomendações

6.1 Conclusões

A utilização de ferramentas CASE educacionais no processo de ensino-aprendizagem em instituições de ensino superior diminui a curva de aprendizagem dos estudantes e possibilita a formação de técnicos profissionais mais capacitados para o mercado de trabalho.

Durante a elaboração do presente trabalho observou-se que o desenvolvimento de ferramentas CASE educacionais pressupõe o uso de uma metodologia adequada, para além da necessidade de uma arquitectura modelo e de uma plataforma de desenvolvimento, específicos.

Chegados ao final do presente trabalho, pesem embora eventuais imperfeições, mormente na profundidade do tratamento teórico, derivado da vastidão das matérias em estudo, contra os limites temporais e de espaço de um trabalho de licenciatura, pode-se afirmar seguramente, que os objectivos propostos inicialmente, foram cumpridos na totalidade.

6.2 Recomendações

Para um trabalho futuro recomenda-se a extensão deste trabalho para o estudo sobre arquitecturas para desenvolvimento de ferramentas CASE educacionais interactivas.

Durante o desenvolvimento do protótipo observou-se que as comunidades de código aberto disponibilizam um vasto ambiente de suporte para a crítica, análise e melhoria no que diz respeito aos requisitos, arquitectura e tecnologias de desenvolvimento de *Software.* Desta forma, recomenda-se o estudo sobre os impactos de desenvolvimento de *Software* educacionais com suporte de comunidades código aberto.

7 Bibliografia

[1] AAFRIN. “Computer Aided Software Engineering Tool’s Classification.” *Aafrim.com.* 14 de Agosto de 2011. http://www.aafrin.com/2011/08/14/computer-aided-software-engineering-tool%E2%80%99s-classification/ (acedido em 31 de Março de 2013).

[2] Amaral, João J. F. *Como fazer uma pesquisar bibliografica.* Ceará, 2007.

[3] Atzeni, Paolo, Stefano Ceri, Stefano Paraboschi, e Riccardo Torlone. *Database Systems - Concepts, Languages and Architectures.* McGraw-Hill Higher Education, 1999.

[4] Baab, L. *Effect of selected factors on students sense of classroom community in distance learning courses.* Dissertação de Doutoramento, Malibu: Universidade de Pepperdine, 2004.

[5] Bass, Len, Paul Clements, e Rick Kazman. *Software Architecture in Practice.* Addison-Wesley, 2012.

[6] Beck, Kent, e Cynthia Andres. *Extreme Programming Explained: Embrace Change.* 2a. Addison-Wesley, 2004.

[7] Beck, Kent, et al. *Manifesto for Agile Software Development.* 2001. http://agilemanifesto.org/ (acedido em 03 de Julho de 2013).

[8] Bergen, Patrick van. “Pipe-And-Filter.” *Garfixia Software Architectures.* 04 de Julho de 2007. http://www.dossier-andreas.net/software_architecture/pipe_and_filter.html (acedido em 18 de Junho de 2013).

[9] Bosch, Jan. *Design and Use of Software Architectures: Adopting and Evolving a Product Line Approach.* Addison-Wesley, 2000.

[10] Campos, Fernanda, Gilda Campos, e Ana Regina Rocha. *Dez etapas para o desenvolvimento de software educacional.* Rio de Janeiro: III Congreso Ibero-Americano de Informática Educativa, 2007.

[11] Cândido, Carlos Henrique. *brModelo: Ferramenta de modelagem conceitual de bancos de dados.* Várzea Grande, 2005.

[12] Costa, António Pedro, Maria João Loureiro, e Luís Paulo Reis. *Development Methodologies For Educational Software: The Practical Case of Courseware SER.* International Conference on Education and New Learning Techonologies, 2009.

[13] Date, Christopher J. *An Introduction to DATABASE SYSTEMS.* 7th Edition. Addison Wesley Longman, 2000.

[14] Elmasri, Ramez, e Shamkant B. Navathe. *Fundamentals of Database Systems.* 4a. Nova York: Addison Wesley, 2003.

[15] Emmerich, Wolfgang, Anthony Finkelstein, e Carlo Montangero. “"The World and the Machine" - A Critical Perspective on Process Technology.” *International Workshop on Research Directions in Process Technology.* Nancy, 1997. 5.

[16] Farias, Adalberto Cajueiro de. *Ferramentas CASE: Suporte, Adoção e Integração.* Monografia, Recife: Universidade Federal de Pernambuco, 2001.

[17] Fernstrõm, C., K. H. Nãrfelt, e L. Ohlsson. “Software factory principles, arcgitecture and experiments.” *IEEE Software* 9 (Março 1992): 36-44.

[18] Filho, Antonio Mendes da Silva. “Arquitetura de Software.” *Engenharia de Software Magazine*, Março 2007: 38-45.

[19] Filho, Wilson P. Paula. *Requirements for an Educational Software Development Process.* Belo Horizonte: Computer Science Dept. - Federal University of Minas Gerais, 2001.

[20] Fuggett, Alfonso. “A classification of CASE technology.” *IEEE Computer Society Press* 26 (Dezembro 1993): 25-38.

[21] Garlan, David, e Dewayne E. Perry. “Introduction to the Special Issue on Software Architecture.” *IEEE Transactions on Software Engineering* 21, n.º 4 (Abril 1995): 269-274.

[22] Ionel, Năftănăilă. “Agile software development methodologies: An overview of the current state of research.” *The Journal of theFaculty of Economics - Economic* 4, n.º 1 (2009): 381-385.

[23] *ISTQB Exam Certification .com.* 02 de Janeiro de 2012. http://istqbexamcertification.com/ (acedido em 06 de Março de 2013).

[24] Jackson, Michael. *The World and the Machine.* Seattle: Association for Computing Machinery, 1995.

[25] Krishnamurthy, Ganesh. “CASE Tools: Adoption and Relevance.” *University of Missouri-St. Louis, St. Louis.* 21 de Janeiro de 2001. http://www.umsl.edu/~sauterv/analysis/F08papers/View.html (acedido em 11 de Março de 2013).

[26] Larman, Craig. *Utilizando Uml e Padrões.* 3ª. Bookman, 2007.

[27] Miller, Granville G. “The Characteristics of Agile Software Processes.” *International Conference and Exhibition on Technology of Object-Oriented Languages and Systems (TOOLS39)*, 2001, 39 ed.: 385-387.

[28] Moreira, Marco Antônio. *Teorias de Aprendizagem.* 2a. São Paulo: Editora Pegagógica e Universitária LTDA, 2011.

[29] National Centre for Technology in Education. *NCTE Advice Sheet – Educational Software.* Dublin: National Centre for Technology in Education, Junho de 2007.

[30] Oliveira, Hamilton, Leonardo Murta, e Cláudia Werner. “Odyssey-VCS: a flexible version control system for UML model elements.” *12th International Workshop on Software Configuration Management.* Lisboa, 2005. 1-16.

[31] Orlikowski, Wanda Janina. *Division Among the Ranks: The social implocations of CASE tools for system developers.* Cambridge, Massachusetts: Massachusetts Institute Of Technology, 1989.

[32] Osorio, Jorge A., Michel R.V. Chaudron, e Werner Heijstek. “Moving From Waterfall to Iterative Development – An Empirical Evaluation of Advantages, Disadvantages and Risks of RUP.” *Euromicro Conference on Software Engineering and Advanced Applications (SEAA)* 37 (2011): 453 - 460.

[33] Pereira, José Luís. *Tecnologia de Bases de Dados.* 3rd Edition. Lisboa: FCA - Editora Informática, 1998.

[34] Pereira, Teresa Sofia Neves Pombo. *Avaliação formativa e aprendizagem da língua portuguesa no contexto de uma comunidade virtual.* Tese de Mestrado, Lisboa: Universidade de Lisboa, 2008.

[35] Rader, Jock, Alan W. Brown, e Edwin J. Morris. *An Investigation into the State of the Practice of CASE Tool Integration.* Relatorio Tecnico, Pittsburgo: Software Engineering Institute, 1993, 40.

[36] Santos, José Alex Soares. “Teorias da Aprendizagem: Comportamentalista, Cognitivista e Humanista.” *Revista Científica Sigma* (Instituto de Ensino Superior do Amapá) 2, n.º 2 (2006): 96-110.

[37] Schwaber, Ken, e Mike Beedle. *Agile Software Development with Scrum.* Prentice Hall, 2001.

[38] Silberschatz, Abrahan, Henry F. Korth, e S. Sudarshan. *Sistema de Banco de Dados.* 5.ª Edição. Campus, 2006.

[39] Silva, Divina Salvador. *O Computador Na Educação.* 14 de Dezembro de 2007. http://www.webartigos.com/artigos/o-computador-na-educacao/3077/ (acedido em 1 de Março de 2013).

[40] Sobrinho, Marcos Fernandes. *Resenha: Teorias de Aprendizagem: Behaviorismo, humanismo e cognitivismo, as antigas teorias behavioristas e cognitivistas.* Resenha, São Paulo: Brasil Escola, 2012.

[41] Sommerville, Ian. *Software Engineering.* 9th. Harlow: Addison-Wesley, 2011.

[42] Teixeira, Adriano Canabarro, e Edemilson Jorge Ramos Brandão. "Software Educacional: O Dificíl Começou." 2003.

[43] Thomas, Ian, e Brian A. Nejmeh. "Definition of Tool integration for Environments." *IEEE Software*, March 1992: 29-35.

[44] Valente, José Armando. "Diferentes usos do Computador na Educação." In *Computadores e Conhecimento: Repensando a Educação*, de José Armando Valente. Campinas: UNICAMP, 1995.

[45] —. *O computador na sociedade do conhecimento.* São Paulo: Editora da Universidade de Campinas, 1999.

[46] van Dam, A. "Visualization research problems in next-generation educational software." *IEEE Computer Graphics and Applications* 25, n.º 5 (Setembro 2005): 88 - 92.

[47] Vannucci, Marco, e Valentina Colla. "Educational software as a learning tool for primary school students." In *New Achievements in Technology Education and Development*, de Safeeullah Soomro, 311-324. Italia: InTech, 2010.

ANEXO

ANEXO A1 : Manifesto Ágil

O Manifesto Ágil é uma declaração de princípios que fundamentam o desenvolvimento ágil de *Software* (Beck, Beedle, et al. 2001).

Manifesto para o Desenvolvimento Ágil de *Software*

Ao desenvolver e ao ajudar outros a desenvolver *Software*, temos vindo a descobrir melhores formas de o fazer. Através deste processo começámos a valorizar:

- Indivíduos e interacções mais do que processos e ferramentas
- *Software* funcional mais do que documentação abrangente
- Colaboração com o cliente mais do que negociação contratual
- Responder à mudança mais do que seguir um plano

Ou seja, apesar de reconhecermos valor nos itens à direita, valorizamos mais os itens à esquerda

Princípios por trás do manifesto ágil

Nós seguimos os seguintes princípios:

- Nossa maior prioridade é satisfazer o cliente, através da entrega adiantada e contínua de *Software* de valor.
- Aceitar mudanças de requisitos, mesmo no fim do desenvolvimento. Processos ágeis se adequam a mudanças, para que o cliente possa tirar vantagens competitivas.
- Entregar *Software* funcionando com frequência, na escala de semanas até meses, com preferência aos períodos mais curtos.
- Pessoas relacionadas à negócios e desenvolvedores devem trabalhar em conjunto e diariamente, durante todo o curso do projecto.
- Construir projectos ao redor de indivíduos motivados. Dando a eles o ambiente e suporte necessário, e confiar que farão seu trabalho.
- O Método mais eficiente e eficaz de transmitir informações para, e por dentro de um time de desenvolvimento, é através de uma conversa cara a cara.
- *Software* funcional é a medida primária de progresso.
- Processos ágeis promovem um ambiente sustentável. Os patrocinadores, desenvolvedores e usuários, devem ser capazes de manter indefinidamente, passos constantes.
- Contínua atenção à excelência técnica e bom *design*, aumenta a agilidade.
- Simplicidade: a arte de maximizar a quantidade de trabalho que não precisou ser feito.
- As melhores arquitecturas, requisitos e *designs* emergem de times auto-organizáveis.
- Em intervalos regulares, o time reflecte em como ficar mais efectivo, então, se ajustam e optimizam seu comportamento de acordo.

ANEXO A2 : Modelo proposto por Elmasri e Navathe

O modelo de dados entidade relacionamento é baseado em uma percepção de um mundo real que consiste numa colecção de objectos básicos, chamados entidades e relações entre esses objectos (Silberschatz, Korth e Sudarshan 2006).

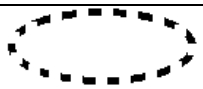
Símbolo	Correspondência	Símbolo	Correspondência
	Entidade forte		Entidade fraca
	Relação não identificadora		Relação identificadora
	Atributo Simples		Atributo identificador
	Atributo Multivalorado		Atributo derivado
	Atributo composto		

Tabela A 2-1: Elementos do modelo entidade relacionamento

(Adaptado de Elmasri e Navate 2003)

ANEXO A3 Casos de uso do Modelador de Base de Dados Relacional

Nome	**CU_RDBM_01:Criar Diagrama**
Actores	Utilizador
Descrição	Inicia quando o utilizador pretende modelar um novo diagrama e solicita a aplicação um novo ambiente para modelar diagramas
Tipo	Primário, Essencial
Referencias	RF-A.2, RF-A.4
Precondições	O programa está aberto
Pós-condições	Diagrama criado

Tabela A 3-1: Descrição do caso de uso CU_RDBM_01

Nome	**CU_RDBM_02: Criar Diagrama Conceptual**
Actores	Utilizador
Descrição	Inicia quando o utilizador pretende modelar um novo diagrama do nível conceptual e solicita a aplicação um novo ambiente para modelar diagramas do nível conceptual
Tipo	Primário, Essencial
Referencias	RF-A.2, RF-A.3
Precondições	O programa está aberto
Pós-condições	Diagrama do nível conceptual criado

Tabela A 3-2: Descrição do caso de uso CU_RDBM_02

Nome	**CU_RDBM_03: Criar Diagrama Lógico**
Actores	Utilizador
Descrição	Inicia quando o utilizador pretende modelar um novo diagrama do nível lógico e solicita a aplicação um novo ambiente para modelar diagramas do nível lógico
Tipo	Primário, Essencial
Referencias	RF-A.4
Precondições	O programa está aberto
Pós-condições	Diagrama do nível lógico criado

Tabela A 3-3:Descrição do caso de uso CU_RDBM_03

Nome	**CU_RDBM_04: Abrir Diagrama**
Actores	Utilizador
Descrição	Inicia quando o utilizador pretende abrir um diagrama e solicita a aplicação para abrir o diagrama
Tipo	Primário, Essencial
Referencias	RF-A.1
Precondições	O programa está aberto
Pós-condições	Diagrama aberto

Tabela A 3-4: Descrição do caso de uso CU_RDBM_04

Nome	**CU_RDBM_05: Aceder Ficheiros de Ajuda**
Actores	Utilizador
Descrição	Inicia quando o utilizador pretende aceder informações de ajuda e solicita para aceder os ficheiros
Tipo	Primário, Essencial
Referencias	RF-A.10
Precondições	O programa está aberto
Pós-condições	Ficheiros disponibilizados

Tabela A 3-5: Descrição do caso de uso CU_RDBM_05

Nome	**CU_RDBM_06: Modelar Diagrama**
Actores	Utilizador
Descrição	Inicia quando o utilizador pretende modelar um dado diagrama e solicita ao programa para alterar o estado do diagrama (adicionando, alterando ou removendo componentes)
Tipo	Primário, Essencial
Referencias	RF-A.5, RF-A.6, RF-A.7, RF-B.6
Precondições	Diagrama aberto
Pós-condições	Estado do diagrama alterado

Tabela A 3-6: Descrição do caso de uso CU_RDBM_06

Nome	**CU_RDBM_07: Adicionar Componentes**
Actores	Utilizador
Descrição	Inicia quando o utilizador pretende adicionar uma componente ao diagrama e solicita ao programa para adicionar a componente ao diagrama
Tipo	Primário, Essencial
Referencias	RF-A.5
Precondições	Diagrama aberto
Pós-condições	Componente adicionada ao diagrama

Tabela A 3-7: Descrição do caso de uso CU_RDBM_07

Nome	**CU_RDBM_08: Editar Componentes**
Actores	Utilizador
Descrição	Inicia quando o utilizador pretende alterar as propriedades de uma componente de um diagrama e solicita ao programa para alterar as propriedades da componente no diagrama
Tipo	Primário, Essencial
Referencias	RF-A.7
Precondições	Diagrama aberto e com no mínimo uma componente seleccionada
Pós-condições	Componente editada

Tabela A 3-8: Descrição do caso de uso CU_RDBM_08

Nome	**CU_RDBM_09: Remover Componentes**
Actores	Utilizador
Descrição	Inicia quando o utilizador pretende remover uma componente do diagrama e solicita ao programa para remover a componente do diagrama
Tipo	Primário, Essencial
Referencias	RF-A.6
Precondições	Diagrama aberto e contendo no mínimo uma componente
Pós-condições	Componente removida do diagrama

Tabela A 3-9: Descrição do caso de uso CU_RDBM_09

Nome	**CU_RDBM_10: Desfazer Operação**
Actores	Utilizador
Descrição	Inicia quando o utilizador pretende desfazer uma alteração feita ao diagrama e solicita ao programa para desfazer a operação
Tipo	Primário, Essencial
Referencias	RF-B.6
Precondições	Diagrama aberto e com no mínimo uma operação realizada
Pós-condições	Operação desfeita

Tabela A 3-10: Descrição do caso de uso CU_RDBM_10

Nome	**CU_RDBM_11: Refazer Operação**
Actores	Utilizador
Descrição	Inicia quando o utilizador pretende refazer uma alteração desfeita ao e solicita ao programa para refazer a operação
Tipo	Primário, Essencial
Referencias	RF-B.6
Precondições	Diagrama aberto
Pós-condições	Operação refeita

Tabela A 3-11: Descrição do caso de uso CU_RDBM_11

Nome	**CU_RDBM_12: Modelar Diagrama de um Exercício**
Actores	Utilizador
Descrição	Inicia quando o utilizador pretende modelar um diagrama de um dado exercício e solicita ao programa para alterar o estado do diagrama (adicionando, alterando ou removendo componentes)
Tipo	Primário, Essencial
Referencias	RF-B.15
Precondições	Programa e exercício aberto
Pós-condições	Diagrama modelado

Tabela A 3-12: Descrição do caso de uso CU_RDBM_12

Nome	**CU_RDBM_13: Gravar Diagrama**
Actores	Utilizador
Descrição	Inicia quando o utilizador pretende gravar as alterações feitas ao diagrama e solicita ao programa para gravar o estado actual do diagrama
Tipo	Primário, Essencial
Referencias	RF-A.8
Precondições	Diagrama aberto e com estado actual não gravado
Pós-condições	Diagrama gravado

Tabela A 3-13: Descrição do caso de uso CU_RDBM_13

Nome	**CU_RDBM_14: Gravar Copia**
Actores	Utilizador
Descrição	Inicia quando o utilizador pretende gravar uma cópia do estado actual do diagrama e solicita ao programa para gravar a copia do estado actual do diagrama
Tipo	Primário, Essencial
Referencias	RF-A.8
Precondições	Diagrama aberto
Pós-condições	Cópia do diagrama gravada

Tabela A 3-14: Descrição do caso de uso CU_RDBM_14

Nome	**CU_RDBM_15: Analisar Diagrama**
Actores	Utilizador
Descrição	Inicia quando o utilizador pretende analisar se um dado diagrama viola regras de modelagem de bases de dados relacionais e solicita ao programa para analisar o diagrama
Tipo	Primário, Essencial
Referencias	RF-A.9, RF-B.1, RF-B.16
Precondições	Diagrama aberto
Pós-condições	Diagrama analisado e resultados disponibilizados

Tabela A 3-15: Descrição do caso de uso CU_RDBM_15

Nome	CU_RDBM_16: Ver Mensagens de Alerta
Actores	Utilizador
Descrição	Inicia quando o utilizador solicita o programa para fazer uma análise de um dado diagrama e o programa detecta erros ou quando o utilizador modifica o diagrama e esse passa a um estado incorrecto
Tipo	Primário, Essencial
Referencias	RF-A.9
Precondições	Diagrama aberto
Pós-condições	Mensagem de erro apresentada

Tabela A 3-16: Descrição do caso de uso CU_RDBM_16

Nome	CU_RDBM_17: Analisar Diagramas de Exercício
Actores	Utilizador
Descrição	Inicia quando o utilizador pretende analisar se um diagrama de um dado exercício viola regras de modelagem de bases de dados relacionais ou se o utilizador pretende saber de o diagrama esta de acordo com o exercício referente e solicita ao programa para analisar o diagrama
Tipo	Primário, Essencial
Referencias	RF-B.16
Precondições	Diagrama aberto
Pós-condições	Diagrama analisado e resultados disponibilizados

Tabela A 3-17: Descrição do caso de uso CU_RDBM_17

Nome	CU_RDBM_18: Converter Diagrama
Actores	Utilizador
Descrição	Inicia quando o utilizador pretende converter um diagrama do nível conceptual a um diagrama do nível lógico ou um diagrama do nível logico a um diagrama do nível conceptual e solicita o programa para converter
Tipo	Primário, Essencial
Referencias	RF-B.3, RF-B.4
Precondições	Diagrama aberto
Pós-condições	Diagrama convertido

Tabela A 3-18: Descrição do caso de uso CU_RDBM_18

Nome	CU_RDBM_19: Imprimir Diagrama
Actores	Utilizador
Descrição	Inicia quando o utilizador pretende imprimir um dado diagrama e solicita ao programa para imprimir o diagrama
Tipo	Primário, Essencial
Referencias	RF-B.8
Precondições	Diagrama existente
Pós-condições	Diagrama impresso

Tabela A 3-19: Descrição do caso de uso CU_RDBM_19

Nome	CU_RDBM_20: Gerar Ficheiro SQL
Actores	Utilizador
Descrição	Inicia quando o utilizador pretende gerar o código SQL de um dado diagrama do nível lógico e solicita ao programa para gerar o código
Tipo	Primário, Essencial
Referencias	RF-B.9
Precondições	Diagrama de nível lógico aberto
Pós-condições	Código gerado

Tabela A 3-20: Descrição do caso de uso CU_RDBM_20

Nome	**CU_RDBM_21: Gravar Imagem**
Actores	Utilizador
Descrição	Inicia quando o utilizador pretende gravar um dado diagrama em formato de imagem e solicita ao programa para gravar o diagrama em formato de imagem
Tipo	Primário, Essencial
Referencias	RF-B.7
Precondições	Diagrama aberto
Pós-condições	Imagem gerada

Tabela A 3-21: Descrição do caso de uso CU_RDBM_21

Nome	**CU_RDBM_22: Editar Exercício Conceptual**
Actores	Utilizador
Descrição	Inicia quando o utilizador pretende alterar os requisitos de um dado exercício de um diagrama do nível conceptual e solicita ao programa para alterar os requisitos
Tipo	Primário, Essencial
Referencias	RF-B.13
Precondições	Exercício aberto
Pós-condições	Estado do exercício alterado

Tabela A 3-22: Descrição do caso de uso CU_RDBM_22

Nome	**CU_RDBM_23: Criar Exercício**
Actores	Utilizador
Descrição	Inicia quando o utilizador pretende elaborar um exercício para modelagem do nível conceptual de uma base de dados relacional
Tipo	Primário, Essencial
Referencias	RF-B.11
Precondições	Programa aberto
Pós-condições	Exercício criado

Tabela A 3-23: Descrição do caso de uso CU_RDBM_23

Nome	CU_RDBM_24: Gravar Exercício
Actores	Utilizador
Descrição	Inicia quando o utilizador pretende gravar o estado actual do exercício e solicita o programa para gravar o exercício
Tipo	Primário, Essencial
Referencias	RF-B.14
Precondições	Exercício aberto
Pós-condições	Exercício gravado

Tabela A 3-24: Descrição do caso de uso CU_RDBM_24

Nome	CU_RDBM_25: Abrir Exercício
Actores	Utilizador
Descrição	Inicia quando o utilizador pretende abrir um dado exercício de modelagem de bases de dados relacionais e solicita o programa para abrir o exercício
Tipo	Primário, Essencial
Referencias	RF-B.12
Precondições	Exercício existente
Pós-condições	Exercício aberto

Tabela A 3-25: Descrição do caso de uso CU_RDBM_25

Nome	CU_RDBM_26: Alterar Propriedades do RDBM
Actores	Utilizador
Descrição	Inicia quando o utilizador pretende alter as configurações do programa a fim de personalizar o mesmo
Tipo	Primário, Essencial
Referencias	Programa aberto
Precondições	Configurações alteradas

Tabela A 3-26: Descrição do caso de uso CU_RDBM_26

Nome	**CU_RDBM_27: Enviar Registo de Erros**
Actores	Desenvolvedor, Utilizador
Descrição	Inicia quando o utilizador detecta um erro no programa e solicita ao programa para enviar uma mensagem com o erro detectado ao desenvolvedor
Tipo	Primário, Essencial
Referencias	RF-B.2
Precondições	Erro detectado
Pós-condições	Mensagem com descrição do erro enviada

Tabela A 3-27: Descrição do caso de uso CU_RDBM_27

ANEXO A4 Interface do utilizador do Modelador de Base de Dados Relacional

1) PRIMEIRA INTERFACE DO PROTOTIPO

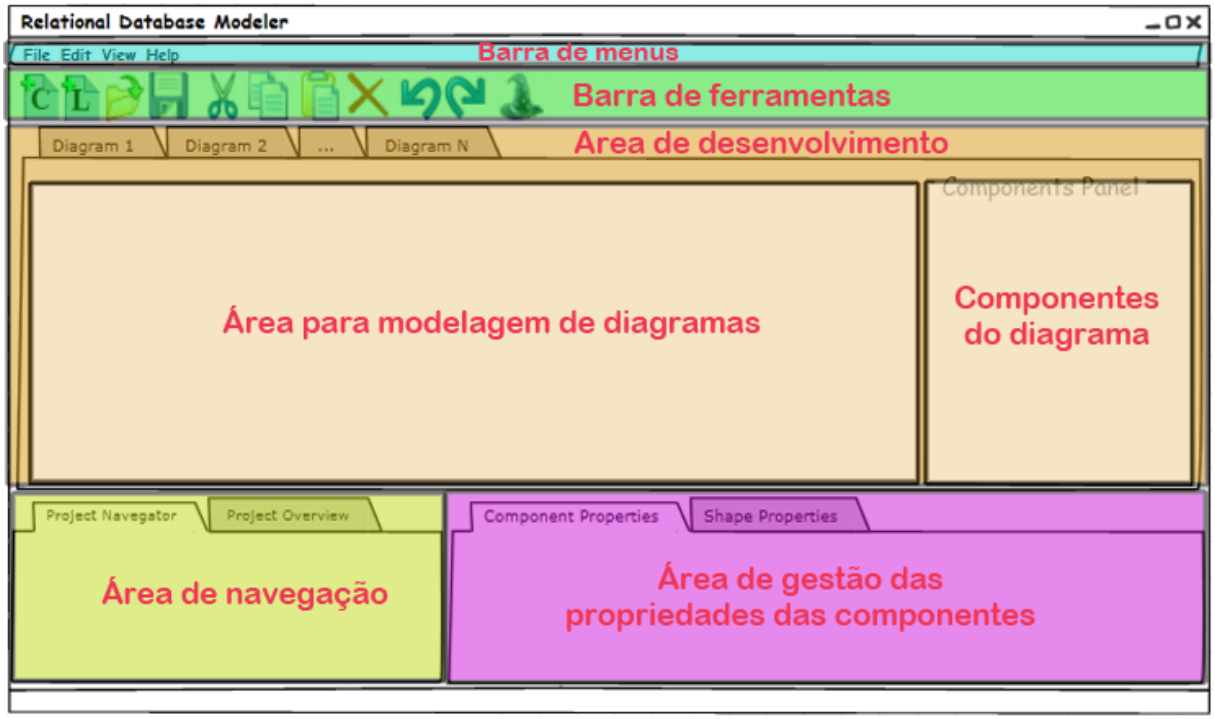


Tabela A 4-1: Primeira interface principal do Modelador de Base de Dados Relacional

2) SEGUNDA INTERFACE DO PROTOTIPO

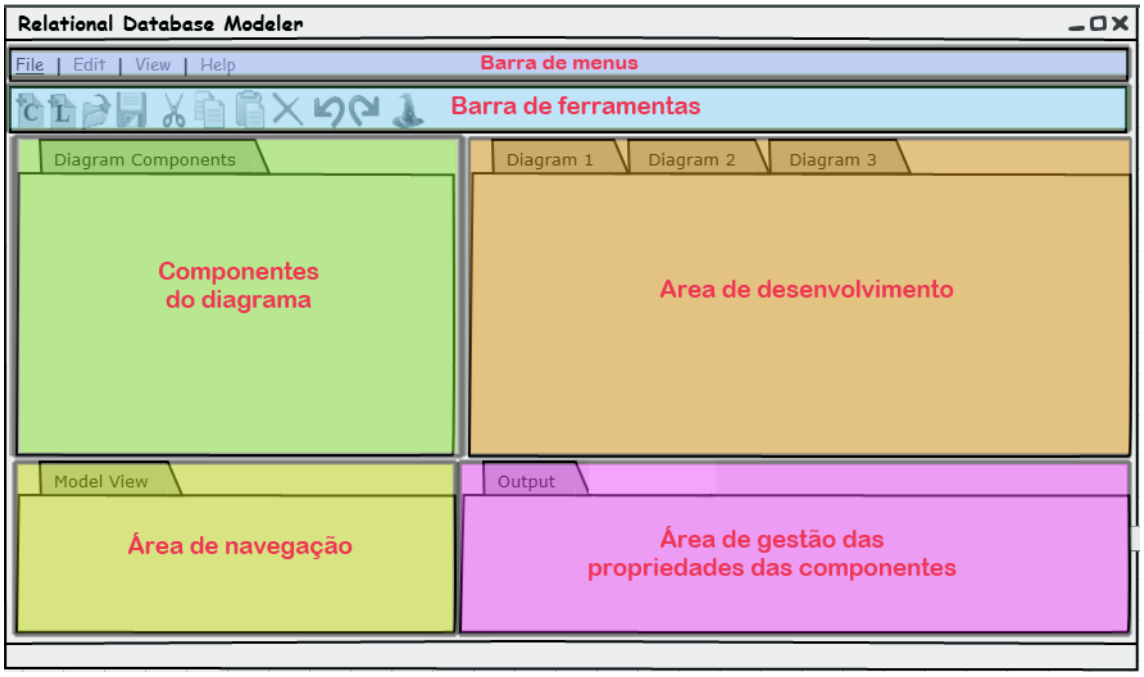


Tabela A 4-2: Segunda interface principal do Modelador de Base de Dados Relacional

3) TERCEIRA INTERFACE DO PROTOTIPO

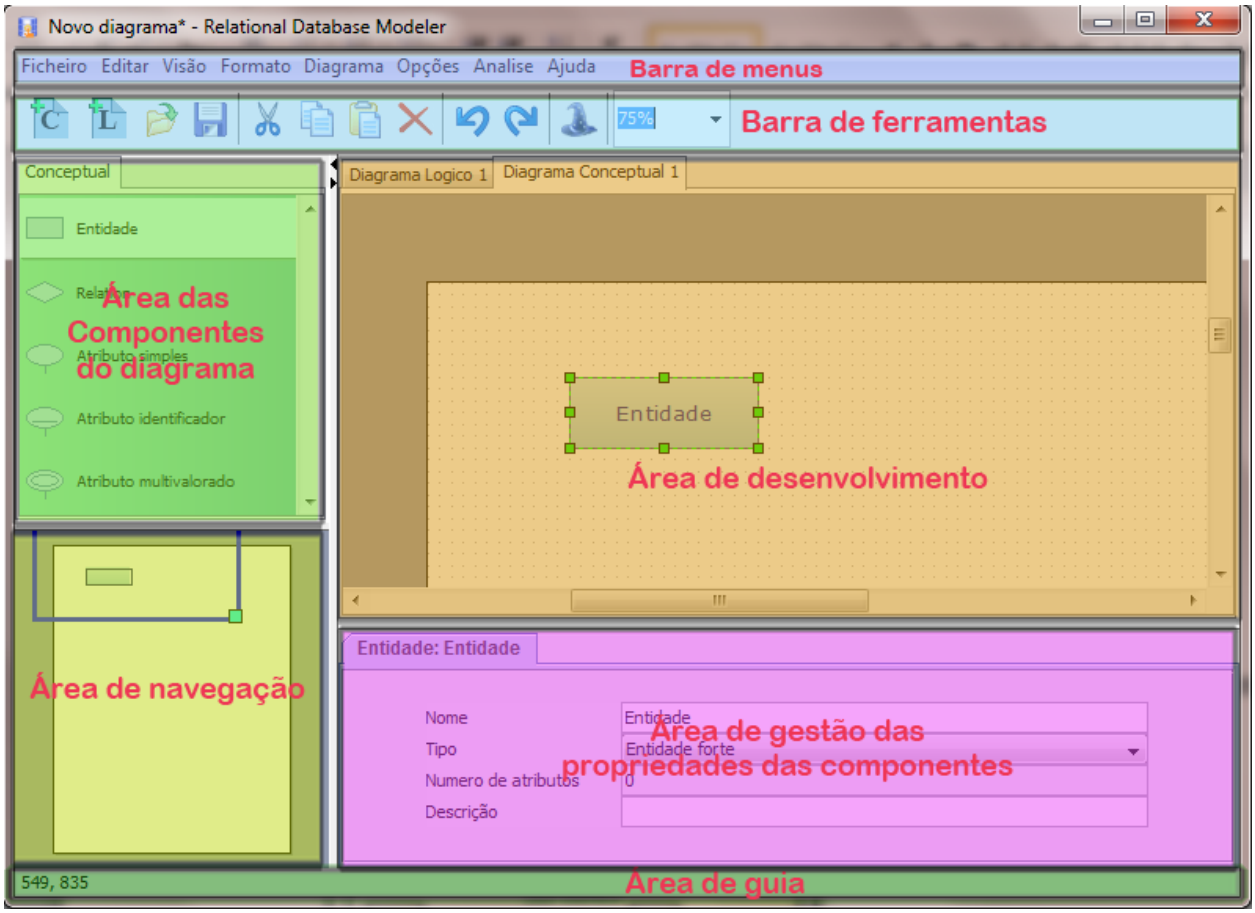


Tabela A 4-3: Terceira interface principal do Modelador de Base de Dados Relacional

ANEXO A5 Código fonte do projecto desenvolvido

Devido a dimensão código fonte do projecto e complexidade para apresentar o mesmo no papel, o código encontra-se no CD a seguir: