RENATO ALBOLEA
RODRIGO CAZELATO PAPETTI

SOFTWARE PARA GERAÇÃO DE MASSA DE TESTES

São Paulo
2006

RENATO ALBOLEA
RODRIGO CAZELATO PAPETTI

nota final 9,1
(nove e hum)

SOFTWARE PARA GERAÇÃO DE MASSA DE TESTES

Trabalho de conclusão de curso apresentado a
Escola Politécnica da Univerdidade de São Paulo
para obtenção do título de Engenheiro Graduado

Área de Concentração:
Engenharia Mecatrônica

Orientador:
Prof. Dr. Marcos Pereira Barretto

São Paulo
2006

FICHA CATALOGRÁFICA

Albolea, Renato
Software para geração de massa de teste / R. Albolea, R.C. Papetti. -- São Paulo, 2006.
83 p.

Trabalho de Formatura - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos.

1.Software (Testes) 2.Java (Projeto) 3.UML I.Papetti, Rodrigo Cazelato II.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos III.t.

RESUMO

Neste relatório, procurare-se expor todos os avanços obtidos na elaboração da documentação, casos de uso, pesquisas bibliográficas e implementação do software para geração de massas de teste. No capítulo de resumo teórico, colocam-se alguns conceitos pesquisados e que servem como base para o desenvolvimento do software. Há uma discussão sobre os tipos de teste mais comuns, como caixa-preta e caixa-branca. Também abordamos algumas linguagens próprias existentes, qualidade de software representada pelo conceito de co-standard e uma discussão um pouco mais genérica sobre o teste em si: seus objetivos, a forma como pode ser encarado, como pode ser conduzido e fatores de sucesso e fracasso para o teste. Evidentemente que toda essa discussão será cercada pelo tema principal de nosso trabalho, ou seja, a geração de massas de teste. Em resumo, propõe-se aqui, desenvolver um software capaz de gerar massas de teste adequadas para condução futura de homologações. Em geral, o teste de um software, em ambiente profissional ou não, se resume a geração de uma massa de teste genérica e verificação de algumas funcionalidades básicas, em especial, aquelas que interagem diretamente com o usuário. Tal metodologia resulta em produtos que atendem parcialmente os requisitos ou apresentam problemas durante sua vida útil. Além disso, há manutenção, retrabalho e descontentamento por parte do usuário. O software proposto fará a verificação de como o alvo do teste lida com as entradas e se as saídas são satisfatórias (teste caixa-preta). Em anexo, temos todos os casos de uso desenvolvidos, já com exemplos de telas,. Casos dentro do escopo deste trabalho são apresentados e discutidos.

Palavras-chave: Engenharia de Software. Testes. Massa de Testes. Java. Casos de Uso.

Abstract

At this report, we try to expose all of the advances concerning documentation, use cases, bibliographic research and program implementation. In the chapter of theorical briefing, there are some concepts investigated that were the base for the software development. We discuss about the two principal types of tests: Black box and White Box. We also discuss about software quality, represented by the concept of co-standard and a more generic approach on software test: Objectives, the way it can be faced, how can we conduct it and which are the factors of success and failure. To sum up, we seek for the creation of a software capable of generating a reliable and relevant mass of test used for homologation. Nowadays, mass of tests are created without professional parameters, just for the test of basic functionalities, especially those related with interface. This behavior results in products that just partially meet users requirements and always have problems during its life time. Furthermore there is always the need for rebuilding and the user getting angry. The appendix presents all the use cases and UML diagrams developed during our work. Cases within the scope are discussed and analyzed.

Key Words: Software Engineering. Tests. Mass of tests. Java. Use Cases

Tabela de Figuras

SUMÁRIO

1. Introdução

O presente trabalho tem por objetivo apresentar alternativas para testes de software, campo vasto e ainda pouco explorado devido a precariedade de estudos e materiais sobre o assunto.

Nosso foco se dará na geração de massa de testes. Atualmente, a geração da massa de testes é feita de modo pouco profissional, por não dizer, amador. Não há nenhum método na seleção dos registros que serão retirados da base e usados no teste. A não ser que haja um ambiente de homologação, os dados são escolhidos, normalmente, por serem os primeiros da tabela. Tal metodologia resulta em produtos que atendem parcialmente os requisitos ou apresentam problemas durante sua vida útil. Além disso, há manutenção, retrabalho e descontentamento por parte do usuário. Mas como garantir que tais dados são realmente relevantes? Como garantir que a carga de registros utilizada é suficiente? Por outro lado, como garantir que a massa de testes gerada não é excessiva e que esforço e dinheiro estão sendo desperdiçados? Enfim como garantir que a massa de testes fornecerá condições para um teste satisfatório?

Neste momento pode-se expandir a discussão. Uma primeira abordagem que deve ser compreendida por desenvolvedores de softwares é o por que do teste do software. A grande maioria dos desenvolvedores acredita que o objetivo do teste de software é provar que o mesmo funciona, mas isso é um grande erro. Ao se tentar verificar que uma simples conta de soma realmente está correta, precisamos verificar se todas as somas

possíveis estão corretas, afim de garantir cem por cento de certeza, o que é simplesmente inviável. Assim, conclui-se que o foco do teste está errado. O verdadeiro objetivo do teste de software não é provar o que funciona corretamente, mas sim encontrar erros, também chamados de Bugs. É neste ponto que uma massa de testes gerada com parcimônia ajuda no teste.

Alguns críticos propalam que o teste de software pode consumir um tempo excessivo do projeto, aumentando de forma desproporcional o custo do mesmo. Porém é comprovado que um teste profissional, orientado, conduzido durante todo o projeto e, principalmente, otimizado, reduz o retrabalho e manutenção diminuindo os custos. Um software de geração de massas de testes caminha lado a lado com este conceito, pois serve como ferramenta para a já citada otimização. A seguir detalhamento do trabalho.

2. Resumo Teórico

No curso de graduação em engenharia tem-se dado muito enfoque em toda a sorte de métodos e processos de cálculo que são essenciais para a realização de projetos de engenharia. Por conseguinte, o aluno tem um grande conhecimento sobre Cálculo Integral, Cálculo Variacional, teorias de controle clássicas e modernas, cálculo estatístico e tantas outras ferramentas úteis para analises numéricas sobre os problemas. Porém, essa estrutura curricular tem como objetivo final formar uma mente "engenheira" .

Apesar dessa estrutura formar profissionais de altíssimo nível, o assunto que discutiremos nesse tópico também é de grande utilidade para um engenheiro moderno, mesmo não possuindo a estrutura clássica supracitada.

Em uma primeira análise, parece que estamos diante de um paradigma de pouca utilidade. Porém, o uso de técnicas não tão focadas em cálculos e mais cálculos tem se mostrado uma ferramenta poderosa na análise de sistemas alto grau de complexidade se os aplicarmos de forma correta em vários âmbitos do mesmo.

A caixa preta consiste de uma máquina com uma ou mais entradas que processa uma ou mais saídas. Ela simplifica muito a análise de um sistema, já pensando em projetos de software, pois a função que ela substitui pode ter qualquer grau de complexidade, desde um simples filtro passa baixa em uma automação industrial, até equações de transferência de calor multidimensionais em um bloco de motor.

A área da engenharia em que essa forma de pensamento é mais utilizada é na eletrônica. Nos primeiros microprocessadores produzidos existiam pouco mais de 2000 transistores para serem polarizados. Por conta disso um engenheiro daquela sabia exatamente qual o numero do transistor deveria ser polarizado para que a lógica que ele necessitava fosse feita.

Com o passar dos anos o número de transistores encapsulados em um circuito cresceu exponencialmente e por conta disso não havia mais como um único profissional ter todos os transistores do circuito integrado em sua memória. Com isso teve-se que criar algumas caixas pretas para simplificar o processo, surgindo as portas lógicas E, OU, OU-EXCLUSIVO e outras para simplificar o trabalho.

Mas, com o tempo, outras caixas pretas tiveram que ser pensadas para abstrair mais o pensamento e com isso formaram-se os flip-flops, Unidade Lógicas Aritméticas(ULA) e assim por diante. Seguindo a mesma

linha de raciocínio, podemos dizer que o grau de abstração na programação chegou hoje às linguagens orientadas a objeto com, por exemplo, o Java, na qual uma simples instrução de soma é na verdade uma caixa preta que comanda diversos operadores lógicos que por sua vez comandam uma serie de transistores para realizar as operações.

Através desses exemplos podemos perceber o quanto já nos utilizamos dessa ferramenta sem perceber e vimos o poder de se olhar por através dela. Nas próximas páginas veremos como podemos utilizar esse conceito, entre outros, na geração de massas de teste.

2.1 Objetivo do teste

Um assunto muito pouco compreendido por desenvolvedores de softwares é o por que do teste do software. A grande maioria dos desenvolvedores acredita que o objetivo do teste do software é provar que o mesmo funciona, mas isso é um grande erro.

Se tentarmos verificar que uma simples conta de soma realmente está correta devemos verificar se todas as somas possíveis estão corretas, o que é simplesmente inviável. Por isso vemos que o foco do teste está errado. O verdadeiro objetivo do software não é provar que ele funciona corretamente, o verdadeiro sentido do teste é encontrar os erros do software, ou seja, os famigerados Bugs.

Essa afirmação nos leva a derrubar mais um conceito sedimentado na maioria dos desenvolvedores que é o de que "Um teste bem sucedido é aquele que não resulta em nenhum erro". A forma correta de se analisar um bom teste é pela quantidade de erros encontrados, e quanto mais erros

melhor! Assim temos que perceber que um teste que não resulte em nenhum erro foi um teste ruim, pois gastou tempo de desenvolvimento e não agregou valor ao projeto.

Devemos também sempre nos lembrar que as características básicas que norteiam os testes são:

- O software deve atender integralmente aos requisitos do cliente
- Um teste completo é impossível
- Deve-se testar primeiramente pequenas funções isoladas, para depois testarmos módulos completos.
- Sempre que possível os testes devem ser projetados e realizados por pessoas não envolvidas em sua programação.

2.2 Tipos de teste

Há uma grande gama de tipos de teste, todos buscando cobrir a maior quantidade de casos de software. Testes de Integração, Testes de Stress, Testes de Função... todos sendo utilizados de acordo com a necessidade do usuário. Porém, há duas estratégias de teste reconhecidamente universais e comumente utilizadas, independente do tipo de teste que se está conduzindo. São elas Teste Caixa-Preta e Teste Caixa-Branca. A seguir um pequeno resumo destas duas estratégias, e como se relacionam com nosso programa.

a) Teste Caixa-Preta: O teste caixa preta consiste em uma verificação de fronteiras do software. O testador fornece as entradas necessárias e espera pela saída, comparando com a esperada. Se há qualquer problema, o usuário pode verificar se a entrada está contaminada. Em caso negativo, o

problema é interno e outros testes deverão ser conduzidos. Nosso software permite ao testador gerar bases de dados para este tipo de teste, basta apenas cadastrar um cenário que contemple apenas as fronteiras.

b) Teste Caixa-Branca: O teste caixa branca é mais invasivo do que o caixa preta, exigindo do testador mais tempo, conhecimento do código e experiência em teste de software. Neste caso, todo o código do programa será varrido. Limites de funções serão testados, relacionamentos, acessibilidade... Em última análise, podemos encarar o teste caixa branca como uma sucessão de "n" testes caixa preta em menor escala. É neste tipo de teste que se encontram erros de lógica e sintaxe, já apontados no teste caixa preta. Para gerar massas de teste para este tipo de aplicação, o usuário deve cadastrar como cenário apenas a função ou conjunto de funções alvo, trabalhando da mesma forma que no caso anterior.

2.3 Particularidades da divisão da programação de softwares

Nos dias de hoje a maior parte dos projetos desenvolvidos são elaborados por mais de uma pessoa ao mesmo tempo, o que é de grande valia quando pensamos que todas as pessoas que estão colaborando unem seus esforços em prol de um objetivo comum. Essa afirmação vale para todas as áreas de atuação, tanto em uma mesa de cirurgia, onde diversos profissionais trabalham de diferentes formas ao mesmo tempo, quanto na elaboração de um projeto cientifico. Mas será que essa mesma afirmação é correta na elaboração de softwares?

É inútil tentar supor que tal afirmação não seja válida na construção de software, pois não há como imaginar como uma única pessoa poderia desenvolver sozinha um sistema com mais de algumas mil linhas de código, que dirá, então, um programa como um sistema operacional que normalmente apresenta algumas dezenas de milhões de linhas de código. Portanto, vamos tentar provar que é possível a união de diversas pessoas na programação de um único software, mostrando abaixo como fazer para que isso seja possível.

O principal problema que uma equipe de programação pode enfrentar durante um projeto é como garantir que toda a equipe escreverá um código que tenha sentido quando unificado e como garantir que a linguagem que uma pessoa está utilizando para escrever seu código segue o mesmo princípio da linguagem utilizada pelo resto da equipe. Para que isso seja possível é necessário que se estabeleça alguns parâmetros básicos antes da equipe sair escrevendo códigos isoladamente, como veremos abaixo:

A)Linguagem Padrão: Uma linguagem padrão especifica a sintaxe e a semântica de uma linguagem de programação (por exemplo Cobol, Ada, C++) ou em uma linguagem de sistemas (por exemplo SQL). Um padrão de linguagem é composto de um conjunto de regras de sintaxe que informam como devemos nomear as variáveis que serão utilizadas, qual será a estrutura de dados que será utilizada, quais mneumônicos serão utilizados dentre outras regras que farão com que cada programador saiba facilmente ler os códigos escritos por seus colegas e saiba como integrar esses códigos com o menor retrabalho possível.

B)Protocolo Padrão: O protocolo padrão especifica para os programadores qual deve ser a estrutura de envio e recebimento de mensagens que deve ser utilizado quando tivermos que fazer dois sistemas diferentes se comunicarem automaticamente, isso evita que cada programador crie seu próprio protocolo o que faria com que em um mesmo sistema diversos protocolos diferentes fossem utilizados aumentando a complexidade do software desnecessariamente e fazendo com que o mesmo programado tivesse que programar tanto o envio das mensagens como o recebimento das mesmas.

C) Bibliotecas Padrões: A estipulação de bibliotecas padrões é necessária para evitar que inconsistências de falta de biblioteca ocorram entre os programadores assim, se estipulam quais bibliotecas poderão ser utilizadas e qual versão das mesmas serão utilizadas. Dessa forma, qualquer recurso que um programador precise e não esteja nas bibliotecas deverá ser escrito como uma função do programa, fazendo com que qualquer programador da equipe possa compilar o programa sem erros dessa ordem.

3. O Software

O software se divide em, basicamente, dois módulos: Telas de cadastro e Núcleo de geração de massas de testes e criação do arquivo de saída. Porém, antes de explicitar o comportamento de cada módulo, cabe ressaltar a estratégia de imlementação utilizada.

3.1 Estratégias de Implementação

Os dois módulos foram construídos baseados no modelo de três camadas, ou seja, classes especializadas que se comunicam por um DataObject. O DataObject consiste em uma classe formada apenas por "getters" e "setters", responsável pelo transporte de dados entre as camadas e com o banco de dados.

As camadas se apresentam da seguinte forma:

A) Camada de Dados: Formada por classes responsáveis pela comunicação com banco de dados, se valendo dos dados do DataObject. Todas as classes que necessitam de informações do banco de dados devem acionar sua respectiva camada de dados.

B) Camada de Negócios: Camada intermediária, possui toda a inteligência do aplicativo, seja em termos de cadastro, seja representando o "core" do programa. Recebe os dados e requisições da camada de interface através do DataObject e os trabalha disparando requisições para a camada de dados, tanto de inserção quanto de consulta, deleção ou alteração.

C) Camada de interface: Essa camada faz toda a interação com o usuário, transmitindo as informações fornecidas por este. Estas informações podem ser dados, como nomes ou valores ou requisições como salvar ou deletar.

As vantagens deste modelo se encontram na garantia da consistência dos dados, que trafegam através de uma classe específica, da garantia de

qualidade do banco de dados que tem seu acesso controlado pela classe de dados e, principalmente, pelo desacoplamento de funções que permite que se façam manutenção ou upgrades em determinadas classes e funções sem necessidade de alteração de todo o código.

3.2 Módulo de Cadastro

O programa encadeia informações de cadastro com dois objetivos. O primeiro é montar uma imagem lógica do banco de dados que o usuário usará como fonte para a massa de testes. O segundo consiste na construção dos cenários de testes pretendidos pelo usuário, entendendo quais são os objetivos e, consequentemente, selecionando os registros mais adequados. A seqüência de cadastro é a seguinte:

1) Cadastro do projeto

   Usuário cadastra o projeto e o caminho para o banco de dados usado.

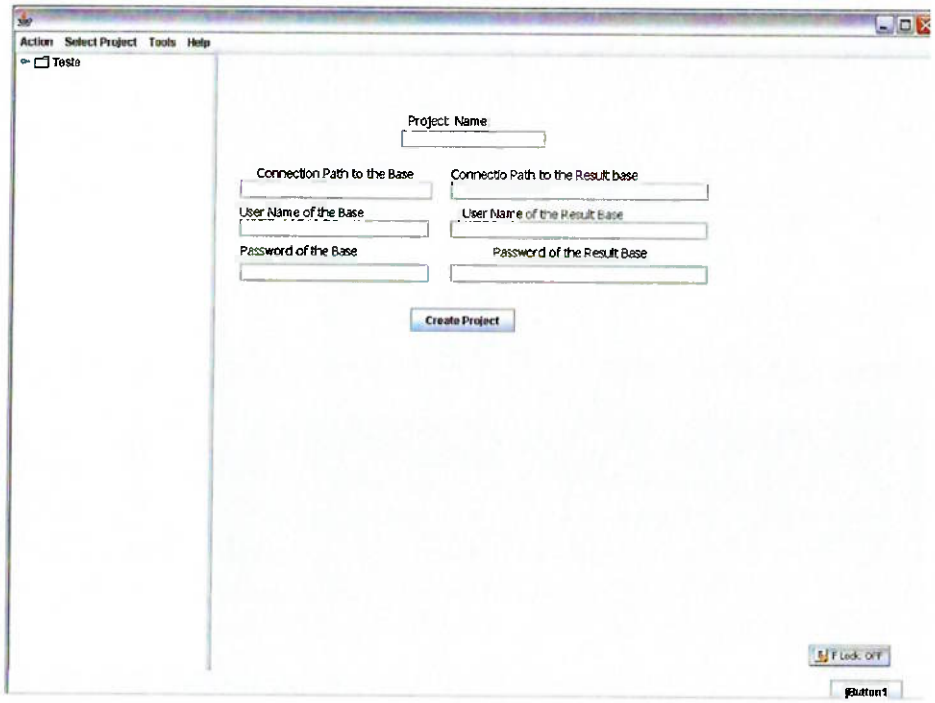

Figura 1 - Tela de cadastro de Projeto

2) Cadastro de Cenário

Usuário cadastra um cenário de testes, que será referência para o tipo de teste conduzido.

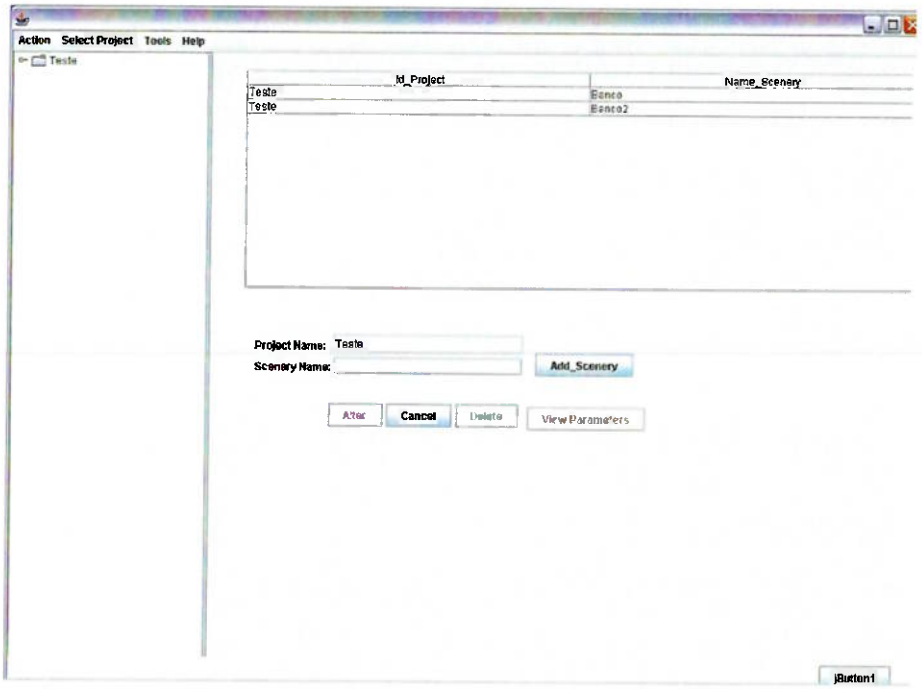

Figura 2 - Tela de cadastro de Cenários

3) Cadastro de Casos

Usuário cadastra os casos pertencentes ao cenário, associados a regra de consulta

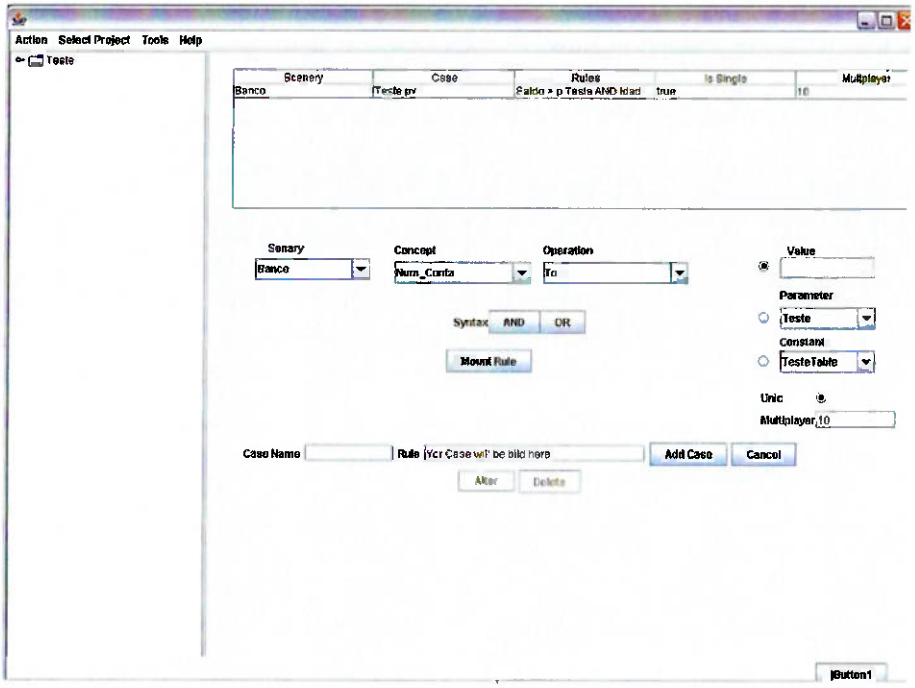

Figura 3 - Tela de cadastro de casos

4) Cadastro de regras

Usuário cadastra regras associadas aos casos, que os especificam e personalizam.

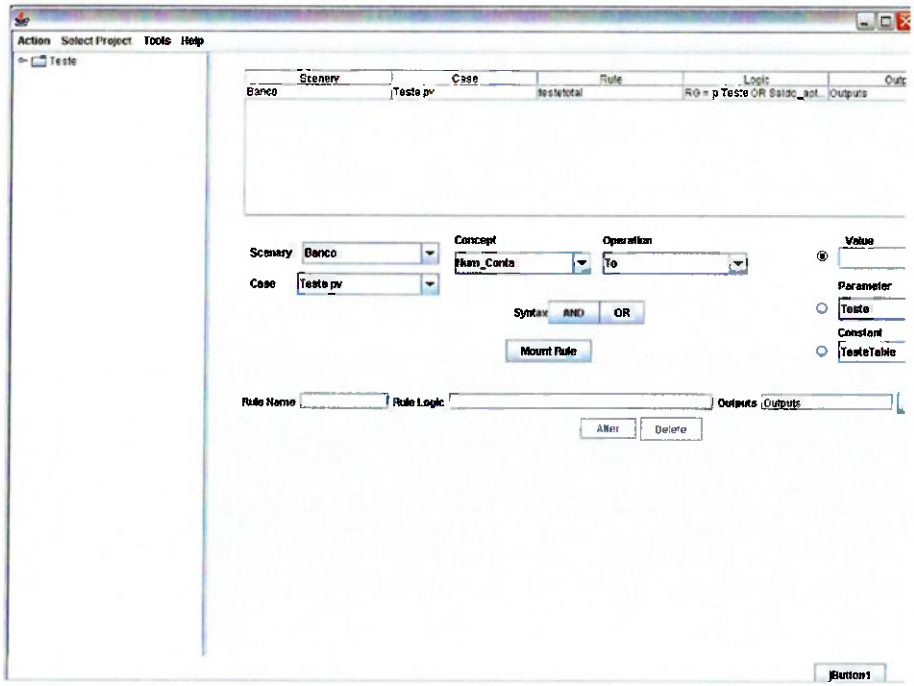

Figura 4 - Tela de cadastro de Regras

5) Cadastro de parâmetros

Usuário associa nomes a conjunto de valores que serão largamente utilizados, poupando trabalho de recadastro.

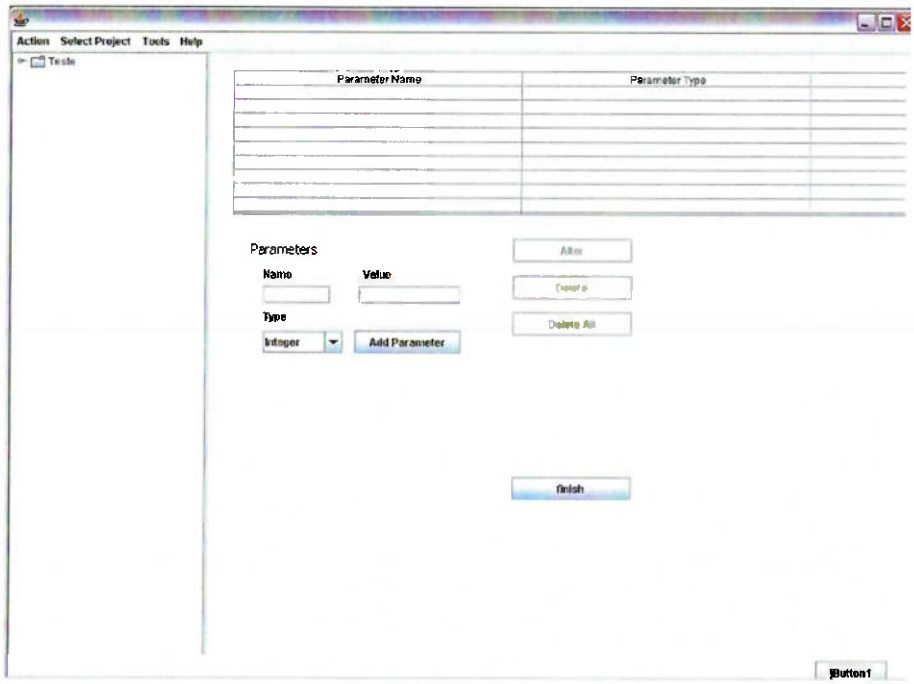

Figura 5 - Tela de cadastro de parâmetro

6) Cadastro de Constantes

Usuário atribui nome a um parâmetro largamente utilizado. Espécie de função Define.

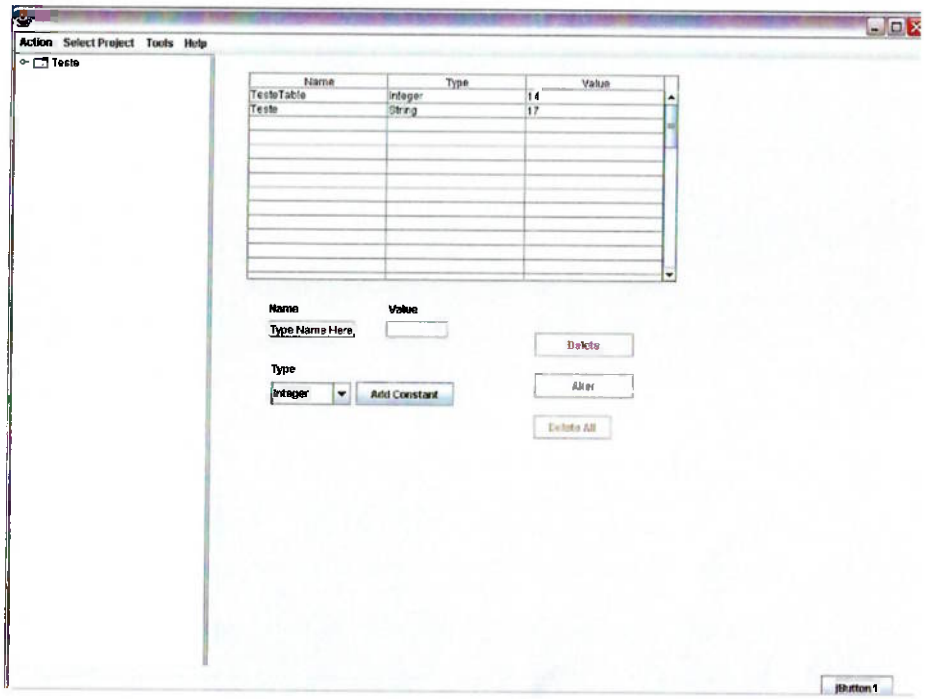

Figura 6 - Tela de cadastro de constantes

7) Cadastro de Tabelas

Usuário descreve lógicamente o banco de dados utilizado como fonte.

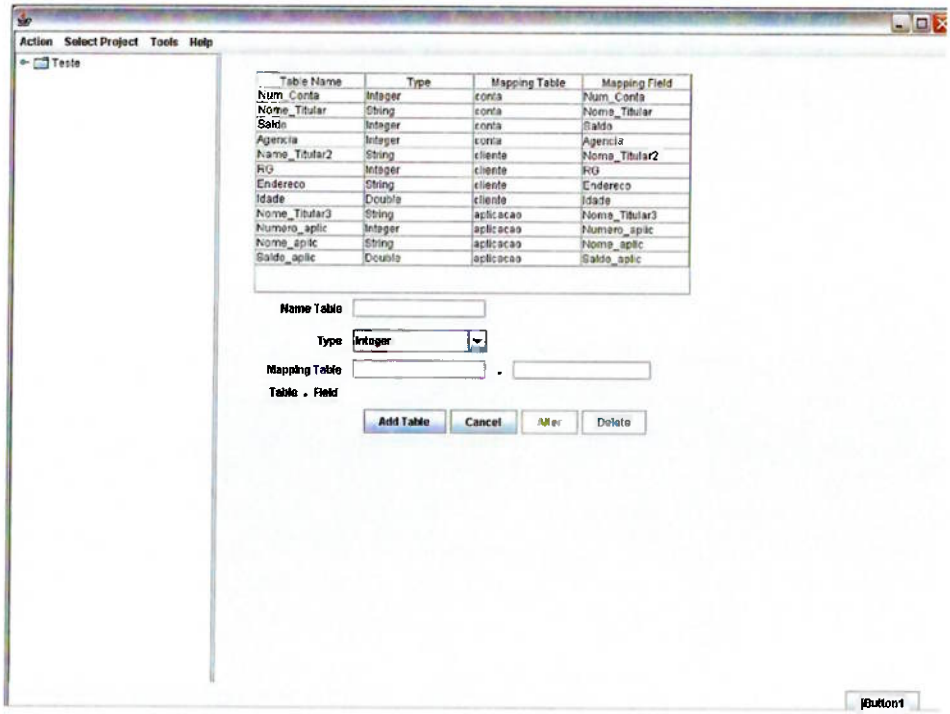

Figura 7 - Tela de cadastro de tabelas

8) Tela de Geração

Permite ao usuário selecionar os cenários a serem considerados na geração da massa de testes.

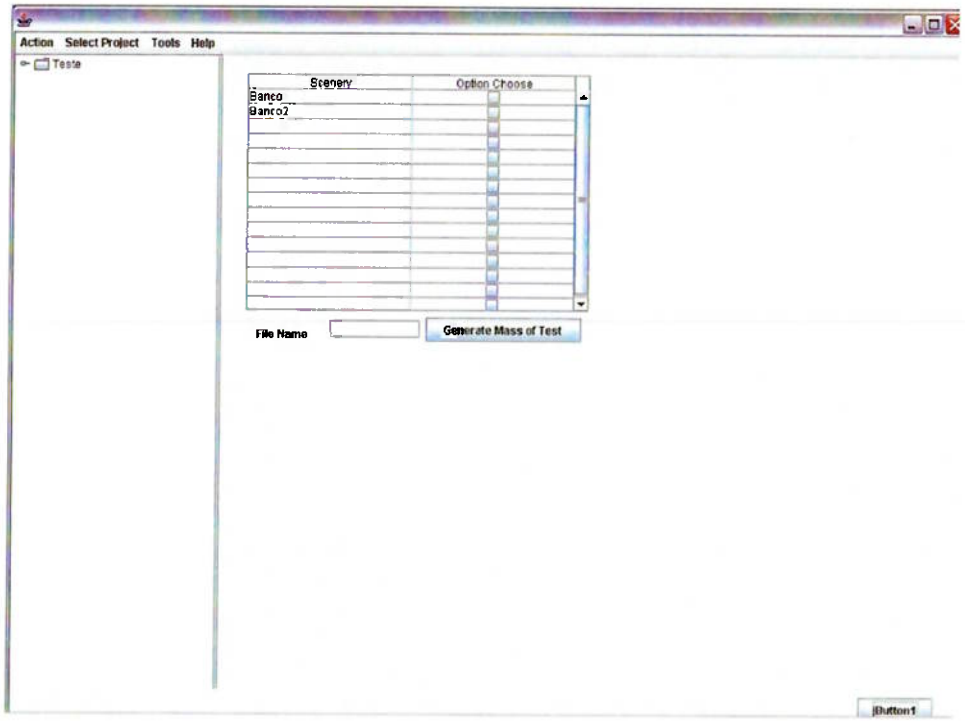

Figura 8 - Tela de Geração

3.3 Modulo de geração de massa de testes

Este é o core do programa. A primeira parte é composta por funções responsáveis pela leitura e quebra das Strings de regras e casos, gerando informações que comporão a consulta. Assim, a regra é quebrada em tabelas, conceitos, valores operações e operadores lógicos sendo gravada em vetores de Strings.

Em seguida, todas as tabelas são reclassificadas no vetor por ordem alfabética e as funções de montagem são chamadas. Essas funções remontam as regras na forma de chamadas sql. Para tanto conectam-se as regras de casos e suas regras personalizadoras através da variável lógica "and".

Com a chamada montada, procedemos com a consulta e os registros encontrados, advindos das regras cadastradas, são gravados em um arquivo txt, que será disponibilizado para o usuário.

Em resumo, o programa tem duas utilidades. A primeira delas é a de orientação na formulação dos cenários de testes. Procura-se automatizar e padronizar o processo, fazendo com que o usuário pense e estude quais cenários são relevantes à sua pesquisa e qual será a estratégia de testes.

Além disso, através de cadastros simples, montam-se consultas complexas na base de dados, facilitando e garantindo que a massa de testes gerada realmente é adequada para aquele determinado caso.

4. Diagramas UML

Apresentam-se aqui os diagramas UML que deram origem ao software, gerados a partir dos casos de uso construídos. Há dois tipos de diagramas: O diagrama de classes, que explicita as classes com seus atributos e métodos, além das interrelações existentes e o diagrama de dados que mostra a lógica e as relações do banco de dados construído para o programa.

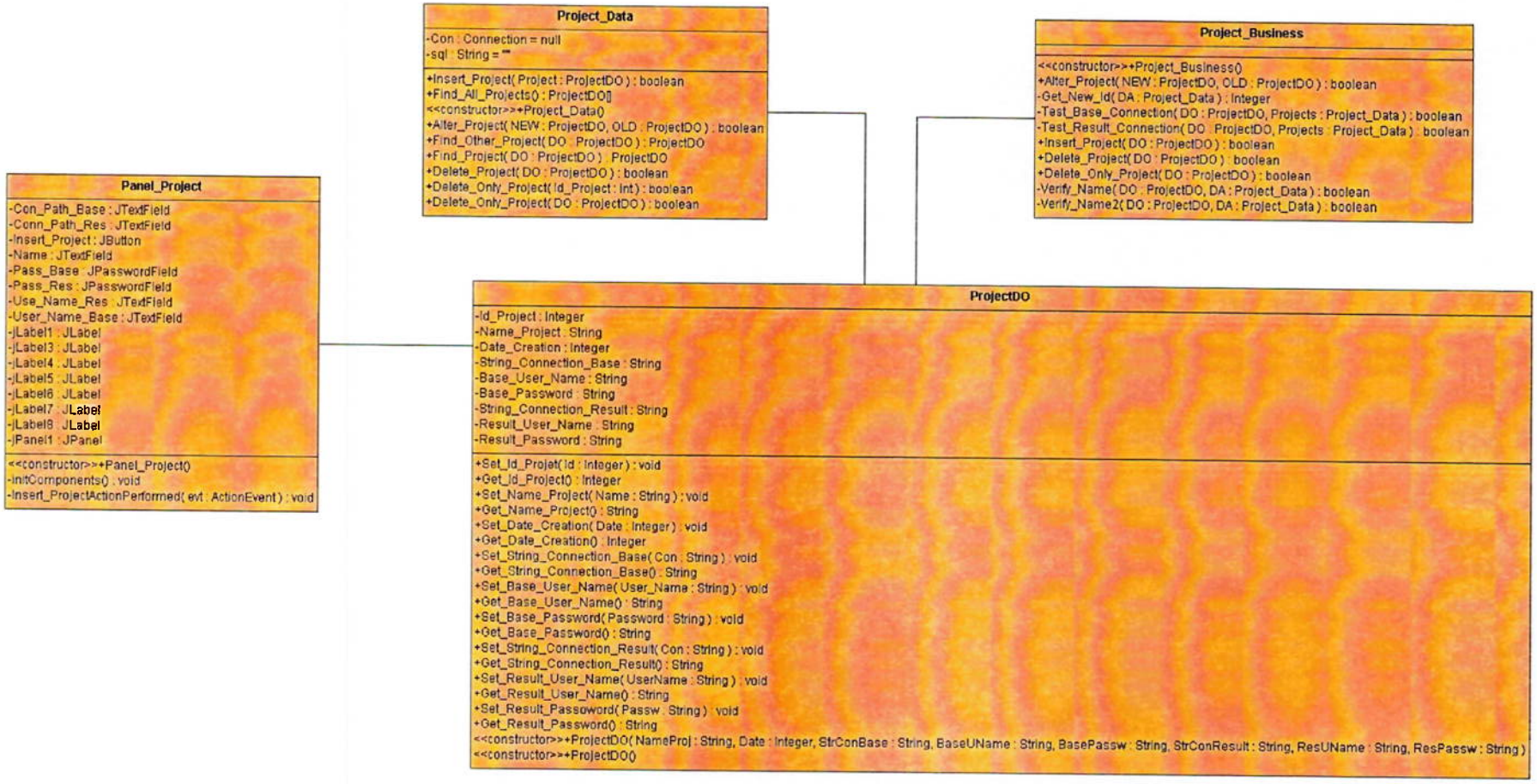

Figura 9 Diagrama UML - Project

Figura 10 Diagrama UML - Constant

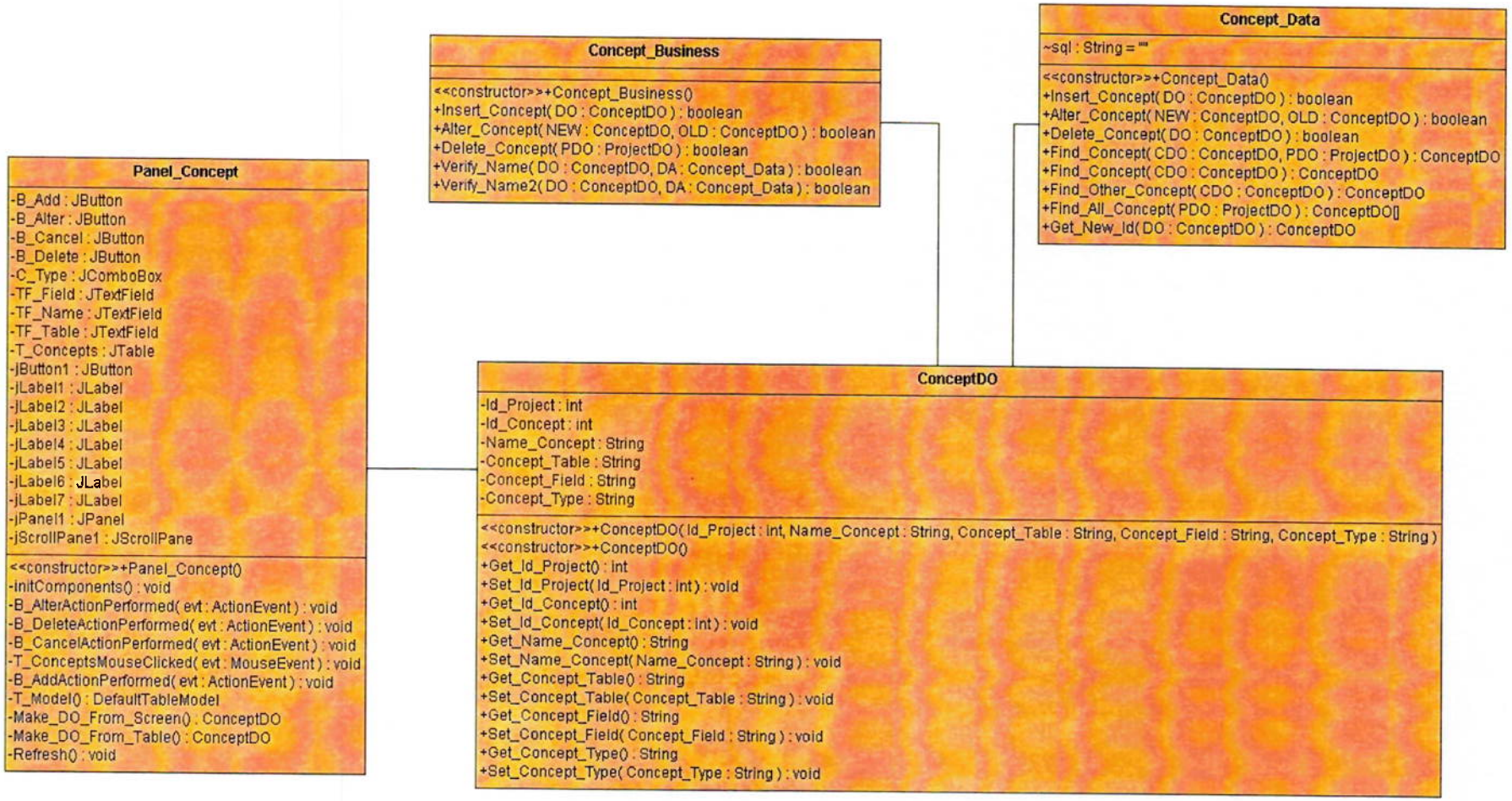

Figura 11 Diagrama UML - Concept

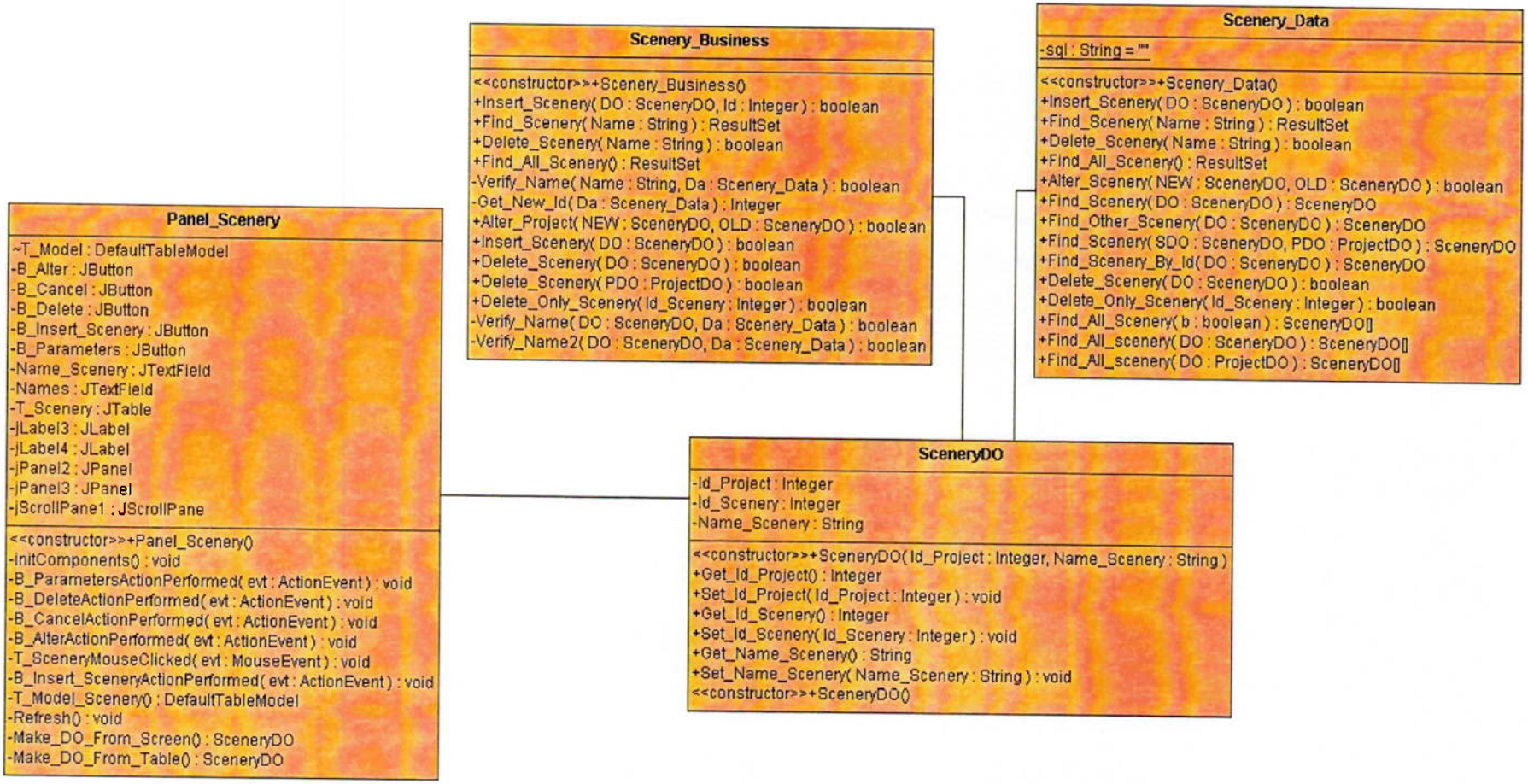

Figura 12 Diagrama UML - Scenery

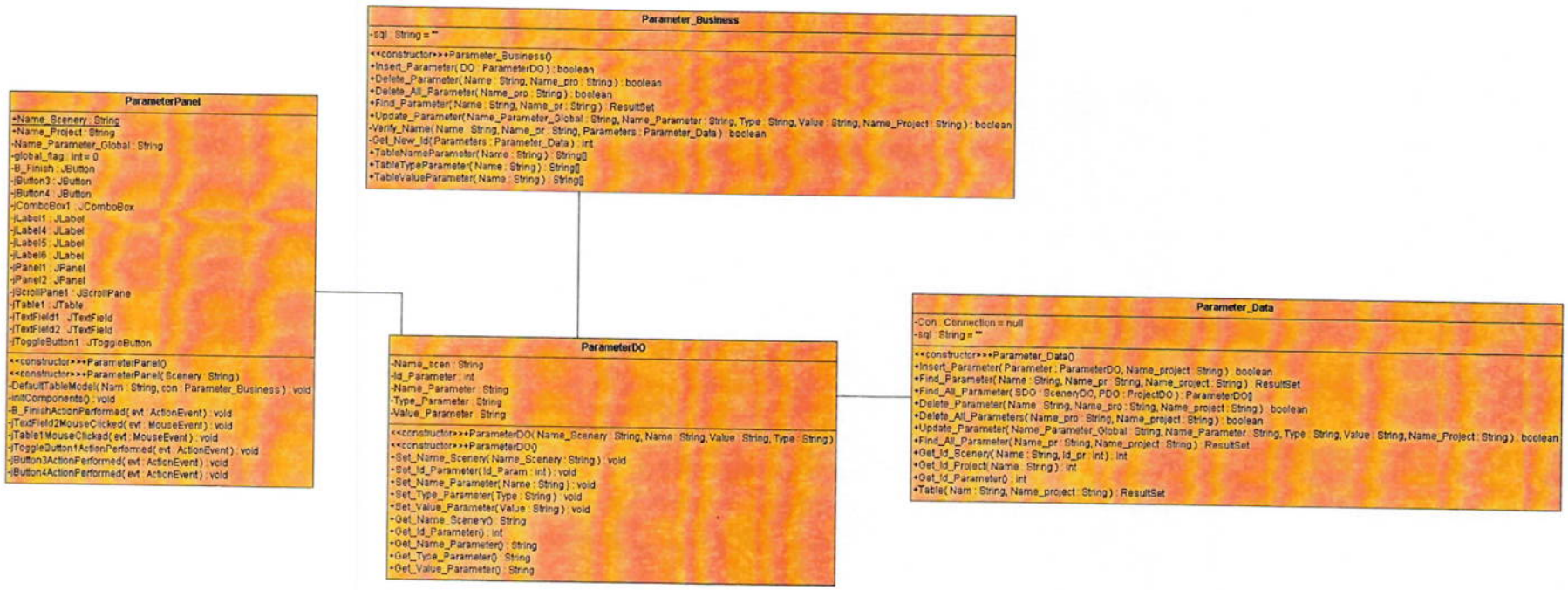

Figura 13 Diagrama UML - Parameter

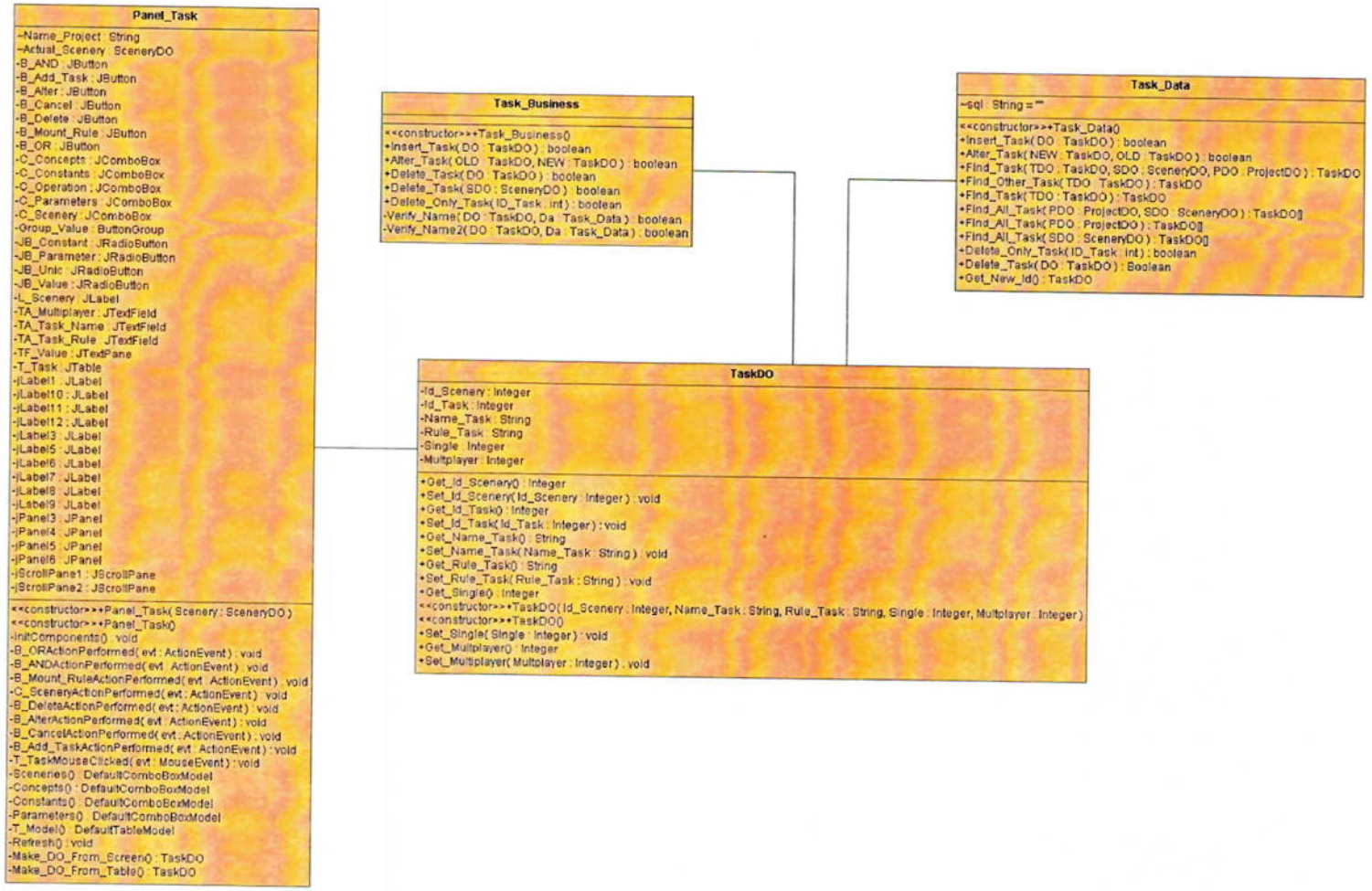

Figura 14 Diagrama UML - Task

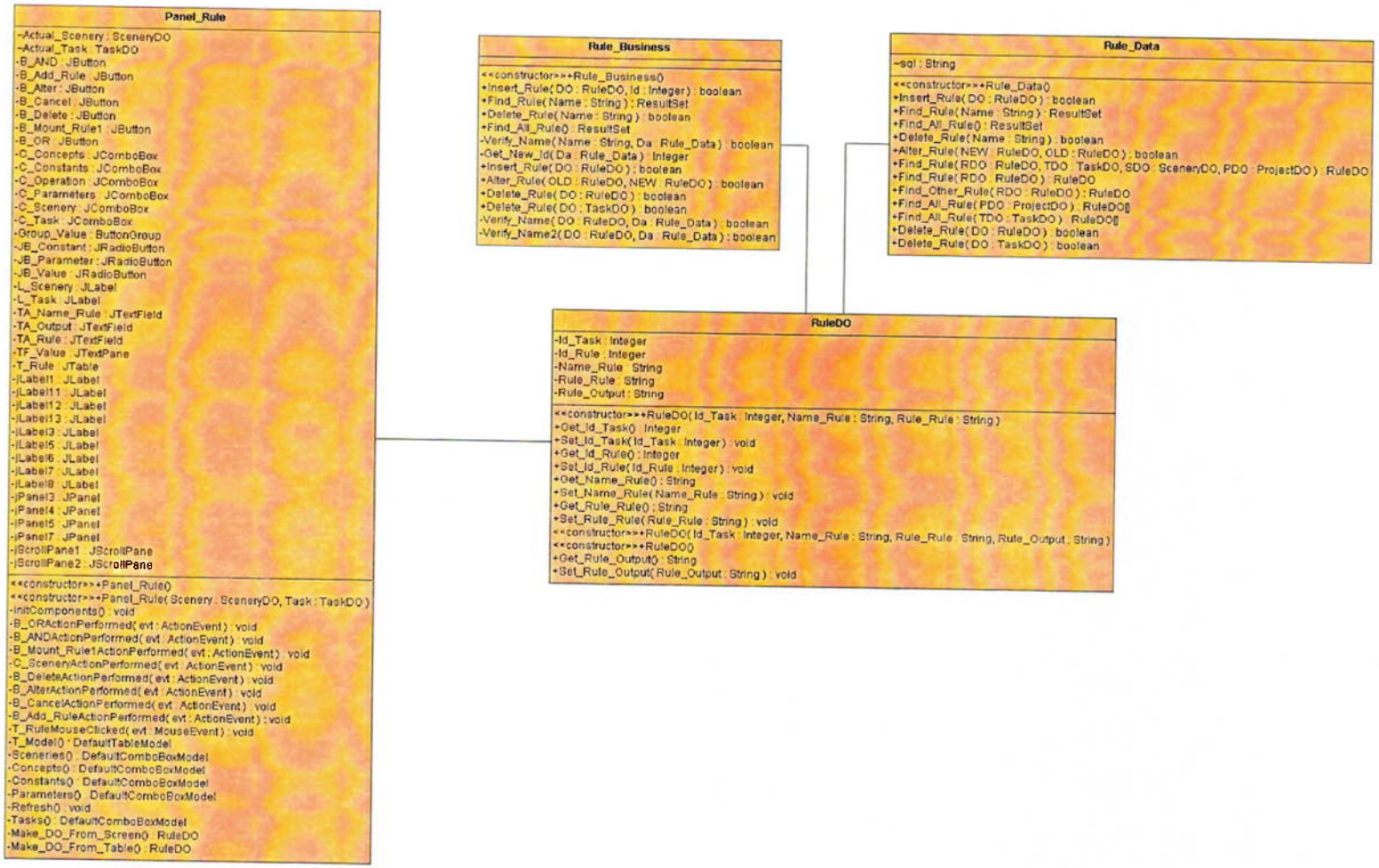

Figura 15 Diagrama UML - Rule

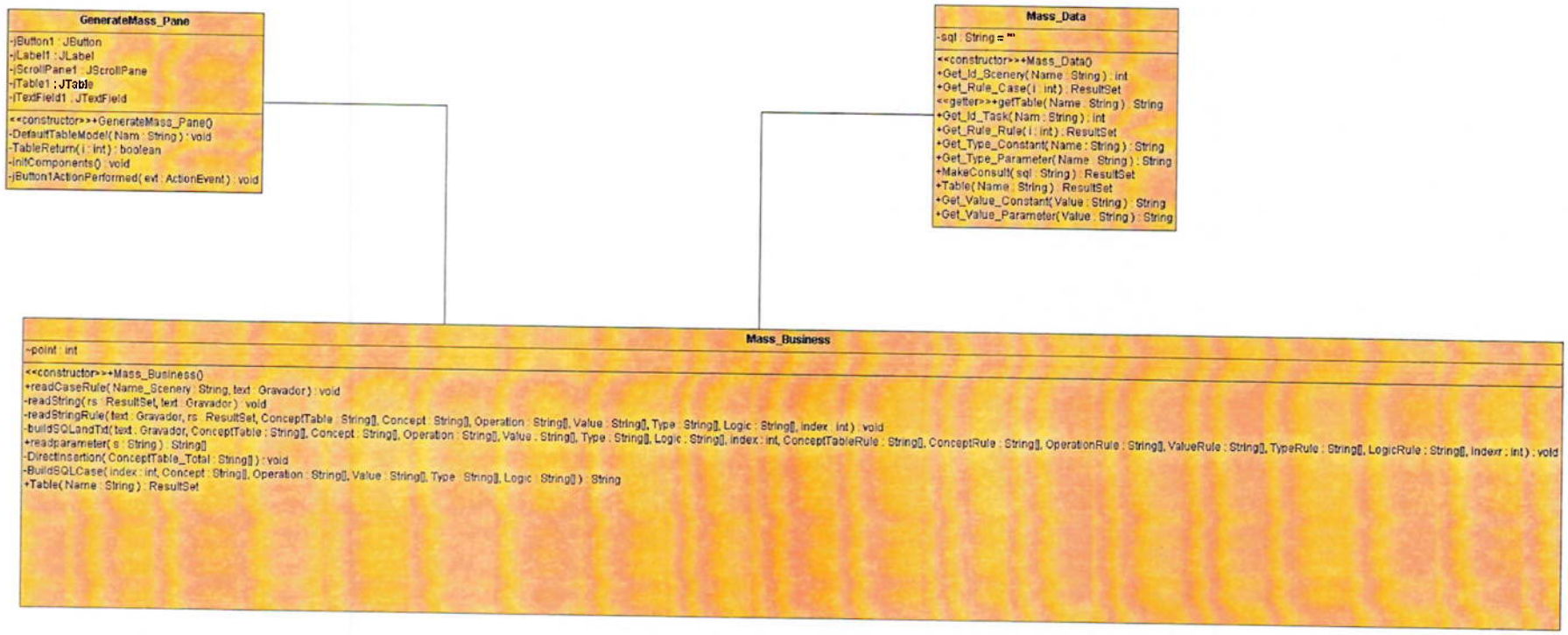

Figura 16 Diagrama UML - Geração da Massa

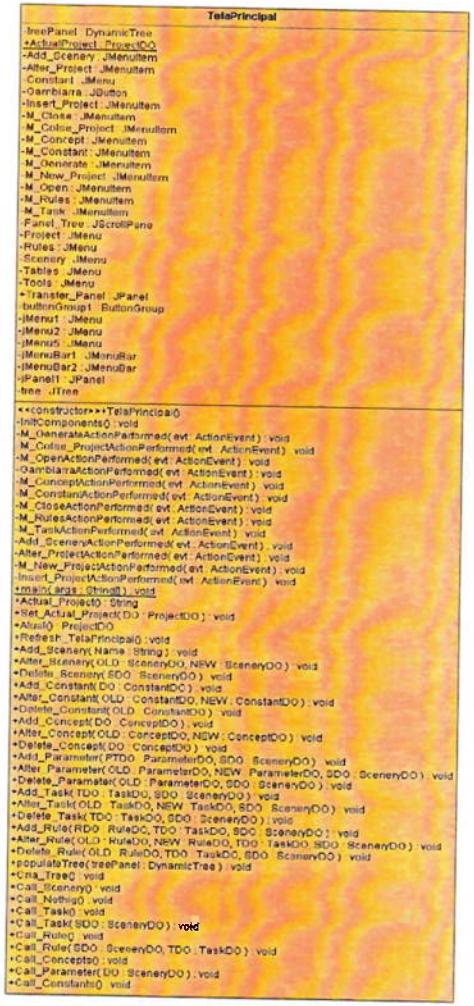

Figura 17 Diagrama UML – Tela Principal

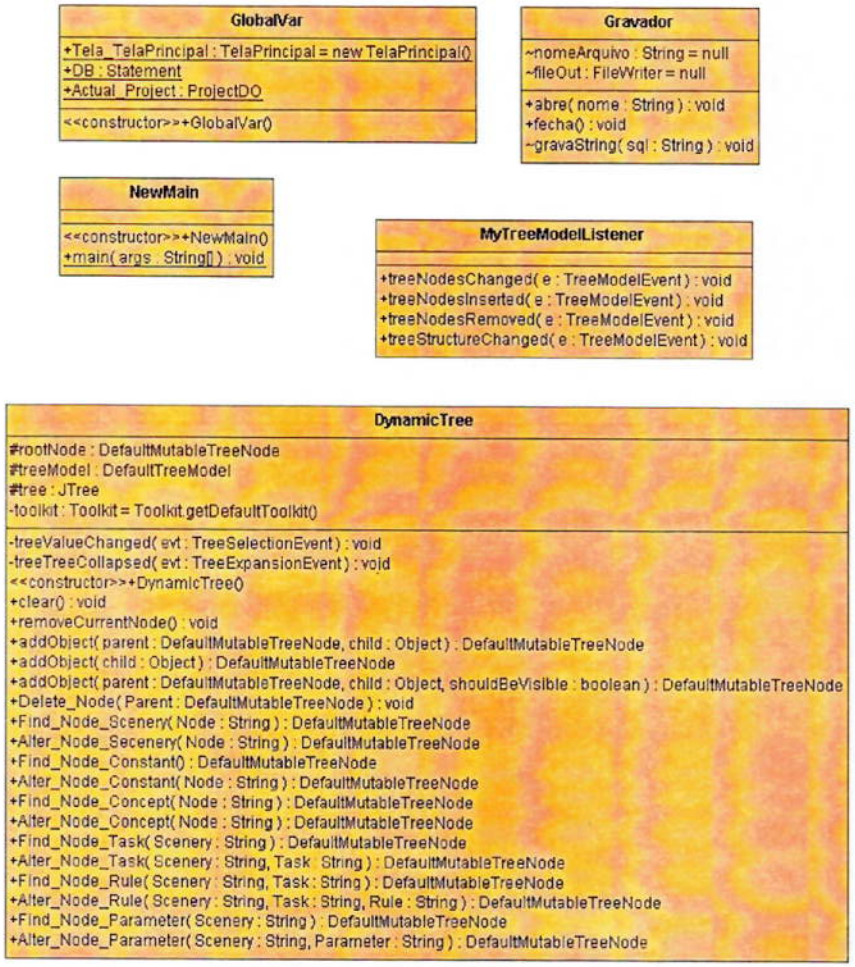

Figura 18 Diagrama UML – Outras Classes

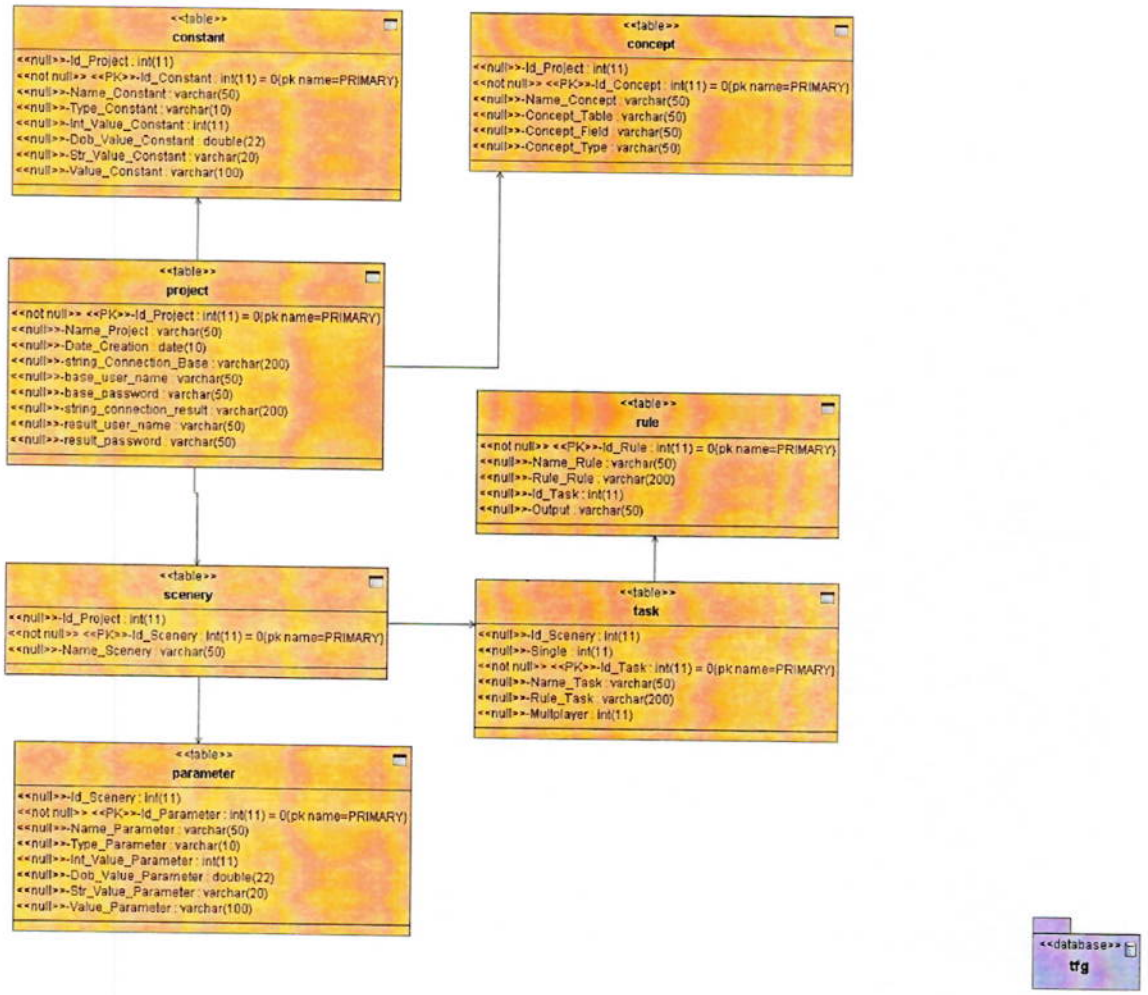

Figura 19 Diagrama UML – Dados

5. Resultados

Como resultado, espera-se que após o cadastro dos cenários, o programa seja capaz de gerar um arquivo txt no formato "CVS", ou seja, exportável para Microsoft Excel, com os registros relevantes para o caso de teste em questão. Para avaliação da eficácia da solução, foi gerada a seguinte base de testes:

```
mysql> describe cliente;
+--------------+--------------+------+-----+---------+-------+
| Field        | Type         | Null | Key | Default | Extra |
+--------------+--------------+------+-----+---------+-------+
| Nome_Titular | varchar(30)  | YES  |     | NULL    |       |
| RG           | int(10)      | YES  |     | NULL    |       |
| Endereco     | varchar(30)  | YES  |     | NULL    |       |
| idade        | decimal(3,0) | YES  |     | NULL    |       |
+--------------+--------------+------+-----+---------+-------+
```

Figura 20 - Tabela do banco de testes

Esta tabela possui os seguintes registros:

```
mysql> select * from cliente;
+-------------+------+----------+-------+
| Nome_Titular | RG   | Endereco | idade |
+-------------+------+----------+-------+
| Paulo       |  123 | Araioses |    11 |
| Pedro       | 1253 | Araioses |    13 |
| Rafael      | 1253 | Araioses |    14 |
| Carlos      | 1253 | Araioses |    15 |
| Jose        | 1253 | Araioses |    16 |
| Renato      | 1253 | Araioses |    17 |
| Marta       | 1253 | Araioses |    18 |
| Eliana      | 1253 | Araioses |    19 |
| Andreia     | 1253 | Araioses |    18 |
| Juliana     | 1253 | Araioses |    20 |
| Clara       | 1253 | Araioses |    21 |
| Luana       | 1253 | Araioses |    22 |
| Camila      | 1253 | Araioses |    23 |
| Silvia      | 1253 | Araioses |    24 |
| Annelise    | 1253 | Araioses |    25 |
| Raquel      | 1253 | Araioses |    26 |
| Andre       | 1253 | Araioses |    27 |
| MArcio      | 1253 | Araioses |    28 |
| Igor        | 1253 | Araioses |    29 |
| Alberto     | 1253 | Araioses |    30 |
| Sandra      | 1253 | Araioses |    31 |
| Debora      | 1253 | Araioses |    32 |
| Renato1     | 1253 | Araioses |    34 |
| Rogerio     | 1253 | Araioses |    35 |
| Valter      | 1253 | Araioses |    36 |
| Ivany       | 1253 | Araioses |    37 |
| Aldo        | 1253 | Araioses |    38 |
| Aracy       | 1253 | Araioses |    39 |
| Luis        | 1253 | Araioses |    40 |
| Helena      | 1253 | Araioses |    41 |
| Renata      | 1253 | Araioses |    42 |
| Miguel      | 1253 | Araioses |    43 |
| Ulisses     | 1253 | Araioses |    44 |
| Tiago       | 1253 | Araioses |    45 |
| Fernando    | 1253 | Araioses |    46 |
| Giovanna    | 1253 | Araioses |    47 |
| Marcela     | 1253 | Araioses |    48 |
| Luiza       | 1253 | Araioses |    49 |
| Rodrigo     | 1253 | Araioses |    49 |
| Maria       | 1253 | Araioses |    50 |
| Anderson    | 1253 | Araioses |    51 |
| Jose        | 1253 | Araioses |    52 |
| Antonio     | 1253 | Araioses |    53 |
| Carolina    | 1253 | Araioses |    54 |
| Tulio       | 1253 | Araioses |    55 |
| Cassio      | 1253 | Araioses |    56 |
| Wagner      | 1253 | Araioses |    57 |
| Nelson      | 1253 | Araioses |    58 |
| Gabriel     | 1253 | Araioses |    60 |
| MArina      | 1253 | Araioses |    61 |
+-------------+------+----------+-------+
```

Figura 21 – Registros

No programa foram inseridos no caso de teste as seguintes regras:

Cenario: Cliente

Caso: Maior de Idade (Constante cadastrada)

Regra_Caso: Idade > 18;

Regra: RG = 1253 (Parâmetro cadastrado)

O arquivo resultante foi:

Your Case was descripted as:

Select * from cliente where idade > 18 and (RG = 1253)

And the significant registers for this case are:

'Nome_Titular','RG','Endereco','idade',

'Eliana','1253','Araioses','19','Juliana','1253','Araioses','20','Clara','1253','Araioses','21','Luana','1253','Araioses','22','Camila','1253','Araioses','23','Silvia','1253','Araioses','24','Annelise','1253','Araioses','25','Raquel','1253','Araioses','26','Andre','1253','Araioses','27','MArcio','1253','Araioses','28','Igor','1253','Araioses','29','Alberto','1253','Araioses','30','Sandra','1253','Araioses','31','Debora','1253','Araioses','32','Renato]','1253','Araioses','34','Rogerio','1253','Araioses','35','Valter','1253','Araioses','36','Ivany','1253','Araioses','37','Aldo','1253','Araioses','38','Aracy','1253','Araioses','39','Luis','1253','Araioses','40','Helena','1253','Araioses','41','Renata','1253','Araioses','42','Miguel','1253','Araioses','43','Ulisses','1253','Araioses','44','Tiago','1253','Araioses','45','Fernando','1253','Araioses','46','Giovanna','1253','Araioses','47','Marcela','1253','Araioses','48','Luiza','1253','Araioses','49','Rodrigo','1253','Araioses','49','Maria','1253','Araioses','50','Anderson','1253','Araioses','51','Jose','1253','Araioses','52','Antonio','1253','Araioses','53','Carolina','1253','Araioses','54','Tulio','1253','Araioses','55','Cassio','1253','Araioses','56','Wagner','1253','Araioses','57','Nelson','1253','Araioses','58','Gabriel','1253','Araioses','60','MArina','1253','Araioses','61'

Em seguida foram inseridos:

Cenario: Cliente
Caso: Meia Idade
Regra_Caso: Idade > 30 (Valor inserido);
Regra: Nome_Titular = Tiago (Constante cadastrada)

O arquivo resultante foi:

Your Case was descripted as:

Select * from cliente where idade > 30 and Nome_Titular = 'Tiago'

And the significant registers for this case are:

'Nome_Titular','RG','Endereco','idade',
'Tiago','1253','Araioses','45',

6. Conclusão

Em vista da escassez de material sobre o tema, acredita-se no pionerismo deste trabalho, bem como na aplicabilidade nos mais variados casos de teste. Bem utilizado, o programa pode apresentar ganho para o testador, tanto em termos de tempo quanto em termos de eficiência. Deixa-se como sugestão para melhorias futuras, a saída do programa como uma planilha Excel ou apenas os ID dos registros, para que o usuário possa se conectar em seu banco de dados pelo próprio programa e buscar os registros. Além disso, pode-se mapear automaticamente o banco de dados do usuário, incluindo as relações entre tabelas, fazendo com que se possa incrementar ainda mais a complexidade das consultas sql.

7. Bibliografia

BARRETTO, M. P., **Notas de Aula – PMR2490**, São Paulo, 2005

BEIZER, Boris. **Software testing techniques**. 2nd ed. New York : Van Nostrand Reinhold, 1990

DE MILO, **Richard A. Software testing and evaluation.** Menlo Park : The Benjamin Cummings Publishin Company, Inc, 1987

HETZEL, William C.**The complete guide to software testing.** 2nd ed. Mass. : QED Information Sciences, 1988

KANER, Cem. **Lessons learned in software testing : a context-driven approach.** New York : Wiley, 2002

MILLER, Edward. **Software testing & validation techniques : tutorial.** 2. Ed. New York : Ieee, 1981

MYERS, Glenford J.**The art of software testing New York**. New York: Editora Wiley. 1979

PERRY, William E. **Effective methods for software testing.** New York : Wiley, 1995

PERRY, William E. **Effective methods for software testing.** 2nd ed. New York : Wiley, 2000

POSTON, Robert M. **Automating specification-based software testing.** Los Alamitos, Calif. : IEEE Computer Society Press, 1996

PRESSMAN, R. S.. **Engenharia de Software**. São Paulo: Bookmark, 2005

ROYER, Thomas C. **Software testing management : lif on the critical path.** Englewood Cliffs : P T R Prentice Hall, 1993

SIEGEL, Shel. **Object oriented software testing : a hierarchical approach.** New York : Wiley Computer Pub., 1996

Apêndice 1

USP::Poli::Mecatrônica/PMR2550

Versão 4.*0*

Histórico das Revisões

Data	Versão	Descrição	Autor
15/06	1.0	Descrição dos casos de Uso após o detalhamento do mesmo.	Renato Albolea Rodrigo Papetti
13/07	2.0	Revisão dos Casos de Uso, após reunião com professor Marcos Barretto	Renato Albolea Rodrigo Papetti
07/09	3.0	Revisão dos Casos de Uso, após implementação parcial do protótipo	Renato Albolea Rodrigo Papetti
20/11	4.0	Revisão dos Casos de Uso após implementação e testes do Protótipo	Renato Albolea Rodrigo Papetti

Índice

Índice de Figura

1 Apresentação

1.1 Objetivo

O objetivo deste documento é a especificação de requisitos para o conjunto de funcionalidades referido como *Sistema de Geração de Massa de Teste*. Nesse documento serão detalhados os casos de uso referentes à criação do Data Generator.

Este sistema irá, a partir de alguns parâmetros fornecidos pelo usuário, indicar quais registros, presentes num banco de dados previamente fornecido, são relevantes para os testes que se seguirão. Assim, o usuário entra com o cenário de testes, o desenho do banco de dados e outros parâmetros, e recebe um arquivo txt com os registros que devem ser usados daquele banco.

2 Conceitos Gerais

2.1 Dicionário de conceitos

Abaixo serão apresentados alguns conceitos que são usados no programa de geração de massa de teste.

- Projeto: No programa é entendido como projeto (Project) um conjunto determinado de tabelas que constituem uma base de dados que será avaliada para a geração da massa de teste.

- Cenário: No programa é entendido como cenário (Scenery) um conjunto de regras, constantes e parâmetros que descrevem de forma macro o caso de teste.

- Caso: No programa é entendido como casos (Case) uma particularização do universo de registros da base de dados que atende a algumas características.

- Regra: No programa é entendido como regras (Rule) uma particularização do universo de registro selecionados no Caso ao qual a Regra pertence.

- Conceito: No Programa é entendido como Conceito (Concept) o campo de uma tabela de um banco de dados

2.2 Tecnologia Utilizada

O servidor de banco de dados que será utilizado para o desenvolvimento e teste do sistema em questão será o MySQL, sendo a interface desenvolvida em linguagem Java através da IDE NetBeans 5.0

2.3 Sistema de Cadastro

O sistema de cadastro é caracterizado por possuir um conjunto de telas que permitirão ao usuário construir todo o cenário em que se desenvolve o teste. Será constituido por n telas construídas para que se possa descrever o banco de dados, como mapear seus registros,quais as principais constantes e parâmetros e, principalmente, quais os casos e as regras do teste. Neste sistema, localizam-se todas as telas de interface com o usuário.

2.4 Sistema de Geração

O sistema de consulta é caracterizado por possuir toda a inteligência do programa. Após todos os dados serm cadastrados de forma bem sucedida, o sistema usa as informações para gerar, automaticamente, consultas sql que procurarão, dentre todos os regsitros presentes na base de dados, aqueles que se adequam aos casos de testes que o usuário cadastrou no sistema.

3 Adicionar Projeto – GMT001

3.1 Breve Descrição

Destina-se ao cadastramento de um novo projeto.

3.2 Atores

Este caso de uso é de uso de todos os usuários do sistema.

3.3 Pré-Condições

O ator deve ter selecionado a opção New Project na aba Action.

3.4 Fluxo de Eventos

Não se aplica.

3.4.1 Fluxo Básico

- O Sistema exibe a tela "New Project" com os campos:
  - ✓ Project Name
  - ✓ Connection path to the base
  - ✓ User name of the base
  - ✓ Password of the base
  - ✓ Connection path to the result base
  - ✓ User name of the result base
  - ✓ Password of the result base

- Quando usuário clicar em Create Project o sistema verifica se o nome do projeto é único e se as strings de conexão fornecidas são validas;
- Se nome do projeto é único e se as conexões com os bancos de dados são validas o sistema grava os dados e constrói uma arvore que contem os elementos relacionados ao projeto conforme se vê na figura abaixo;
- Se não sistema exibe uma mensagem pedindo ao usuário para entrar com um nome novo ou corrigir a string de conexão;

A Tela new project é a seguinte:

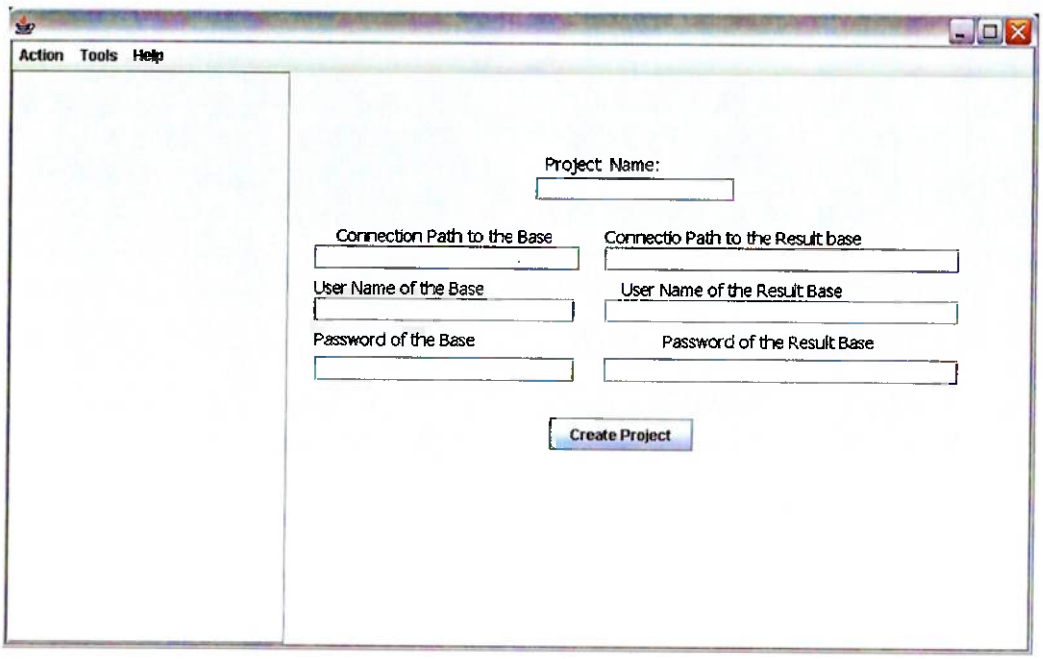

Figura 3.1 Tela de cadastro de Projeto(New Project)

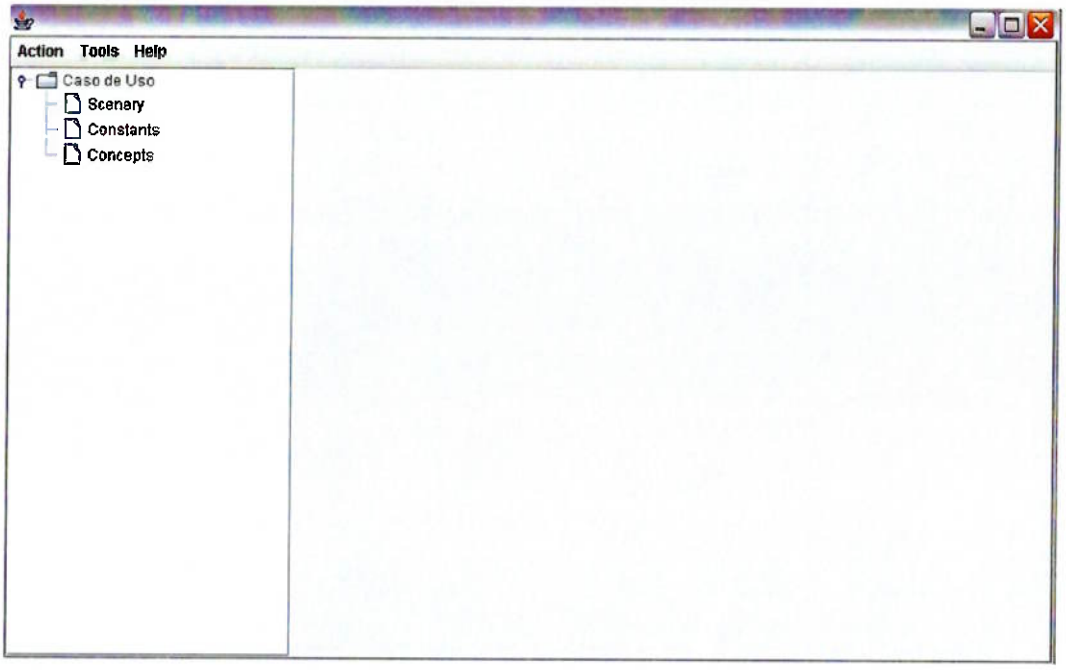

Figura 3.2 Árvore do Projeto

3.4.2 Fluxos Alternativos

Não aplicavel

3.4.3 Requerimentos Especiais

Não aplicável.

3.4.4 Pós-Condições

Após a criação do novo projeto, qualquer outro projeto que estiver aberto será fechado automaticamente.

3.4.5 Pontos de Extensão

Não Aplicável.

4 Adicionar, Alterar, Excluir Constantes – GMT002

4.1 Breve Descrição

Destina-se a permitir o cadastro de constantes por parte do usuário.

4.2 Atores

Este caso de uso é de uso de todos os usuários do sistema.

4.3 Pré-Condições

Para que se possa cadastrar uma constante é necessário haver um projeto corretamente cadastrado.

4.4 Fluxo de Eventos

Não se aplica.

4.4.1 Fluxo Básico

- O sistema exibe a tela de cadastro de constantes com os botões "Alter" e "Delete" desabilitados e os seguintes campos:
  - ✓ Name
  - ✓ Value
  - ✓ Type (Esse campo só pode assumir os valores Integer, Double e String)

- Quando o usuário clica no botão "Add Constant" o sistema verifica se o nome da constante é único e se o tipo determinado pelo usurário é valido;
- Se o nome for único e o valor fornecido for valido o sistema atualiza a arvore do programa e grava os dados;
- Se não for único o sistema exibe uma tela pedindo ao usuário para inserir um novo nome ou informando que o tipo da constante selecionado não é compatível com o valor fornecido;

A tela de cadastro de constante é a seguinte:

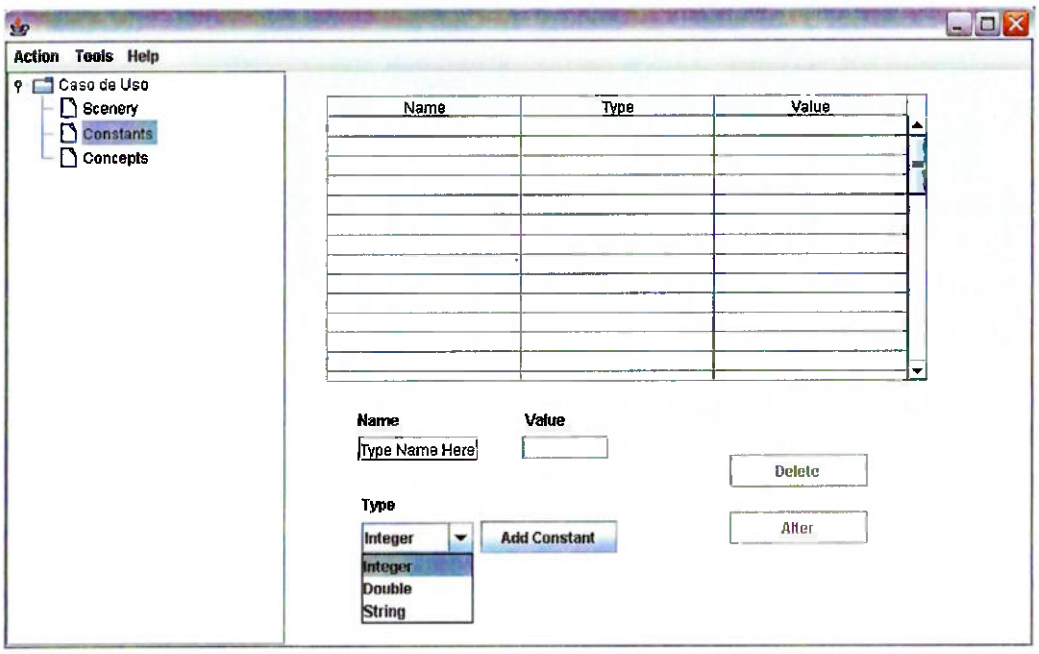

Figura 4.1 Tela de cadastro de Constante(Add Constant)

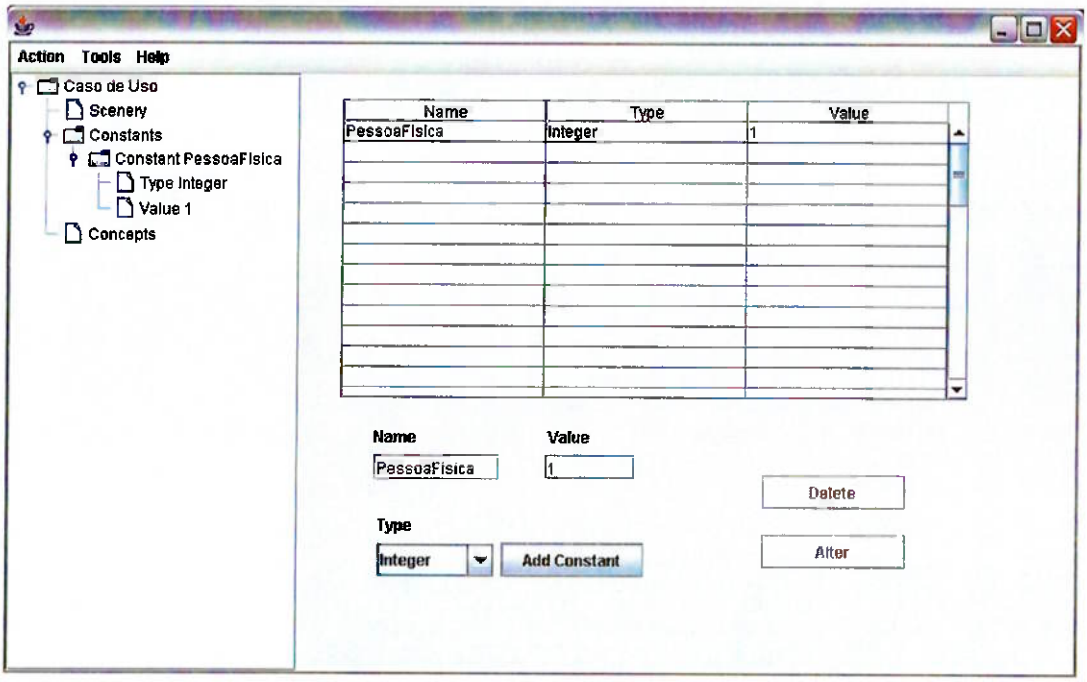

Figura 4.2 Tela após a ainclusão da Constante

4.4.2 Fluxos Alternativos

A)Exclusão de uma constante

- O ator seleciona uma constante já cadastrada com na tabela de constantes
- O sistema habilita os botões "Alter" e "Delete" e todas as informações sobre a constante são copiadas para os seus respectivos campos de criação;
- Se o ator clicar no botão "Delete" o sistema exibe uma mensagem padrão pedindo a confirmação da exclusão;
- Se o ator confirmar a exclusão a tela é atualizada;

- Se o ator não confirmar a exclusão o processo é abortado e nenhuma modificação é feita;

B)Alteração

- O ator seleciona uma constante já cadastrada com na tabela de constantes
- O sistema habilita os botões "Alter" e "Delete" e todas as informações sobre a constante são copiadas para os seus respectivos campos de criação;
- Quando o usuário clica no botão "Alter" o sistema verifica se o nome da constante é único e se o tipo determinado pelo usurário é valido;
- Se o nome for único e o valor fornecido for valido o sistema atualiza a arvore do programa e grava os dados;
- Se não for único o sistema exibe uma tela pedindo ao usuário para inserir um novo nome;
- Se ator clicar no botão "Cancel" a operação de alteração é abortada e nenhuma modificação é feita.

4.4.3 Requerimentos Especiais

Não se aplica.

4.4.4 Pós-Condições

Não se aplica.

4.4.5 Pontos de Extensão

Nenhum.

5 Adicionar, Alterar, Excluir Conceito– GMT003

5.1 Breve Descrição

Destina-se ao cadastro de conceitos ao programa para que seja criada uma réplica da estrutura do banco de dados do usuário.

5.2 Atores

Este caso de uso é de uso de todos os usuários do sistema.

5.3 Pré-Condições

Para que se possa cadastrar um conceito é necessário haver um projeto corretamente cadastrado.

5.4 Fluxo de Eventos

Não se aplica.

5.4.1 Fluxo Básico

- O sistema exibe a tela de cadastro de Conceito com os botões "Alter" e "Delete" desabilitados e com os seguintes campos:
  - ✓ Concept Name

- ✓ Type(Esse campo só pode assumir os valores Integer, Double e String)
- ✓ Mapping: Table
- ✓ Mapping: Field

- Quando o usuário clica no botão "Add Concept" o sistema verifica se o nome do conceito é único no projeto;
- Se o nome for único o sistema atualiza a tela;
- Se não for único o sistema exibe uma tela pedindo ao usuário para inserir um novo nome;

A tela de criação de conceitos tem a seguinte forma:

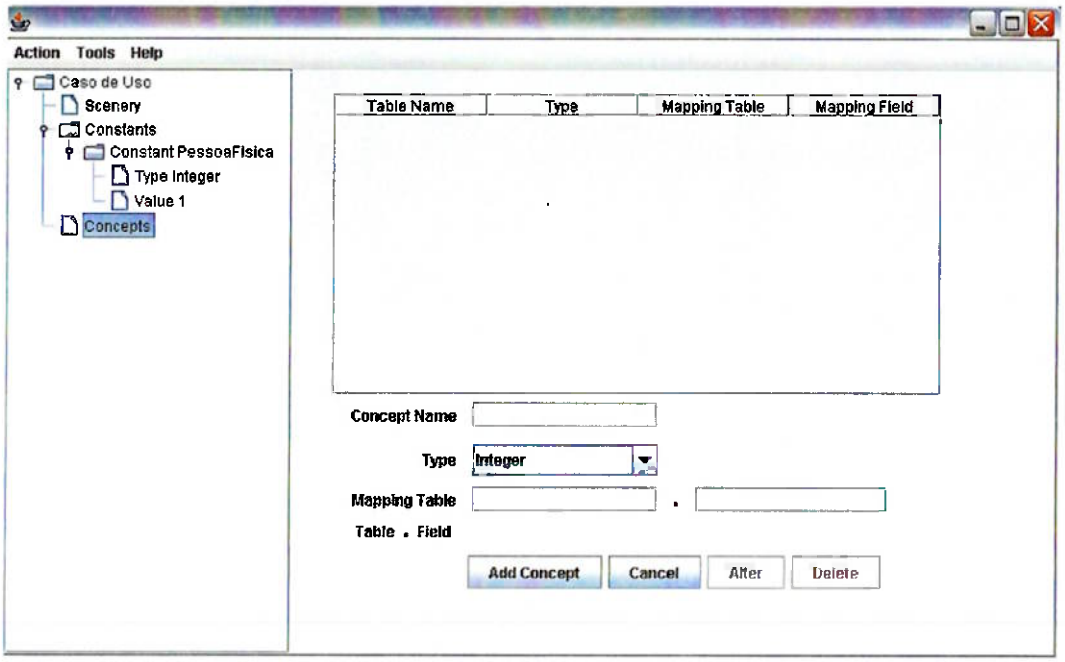

Figura 5.1 Tela de cadastro de Conceito(Add Concept)

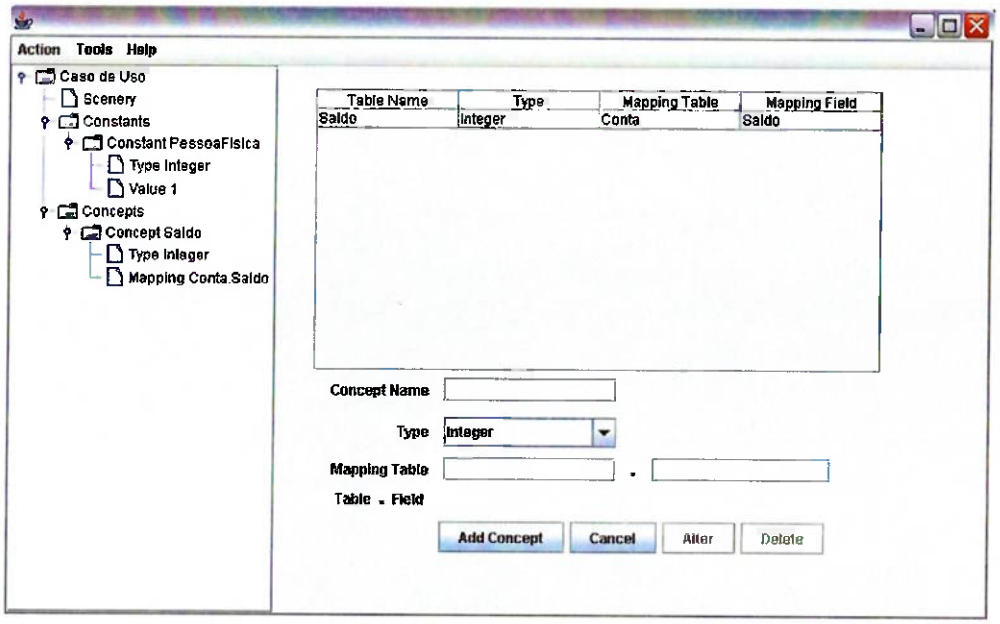

Figura 5.2 Tela após a inclusão de um Conceito

5.4.2 Fluxos Alternativos

A)Exclusão de um conceito

- O ator seleciona um conceito já cadastrado na tabela de Conceitos
- O sistema habilita os botões "Alter" e "Delete" e todas as informações sobre o conceito são copiadas para os seus respectivos campos de criação;
- Se o ator clicar no botão "Delete" o sistema exibe uma mensagem padrão pedindo a confirmação da exclusão;
- Se o ator confirmar a exclusão a tela é atualizada;
- Se o ator não confirmar a exclusão o processo é abortado e nenhuma modificação é feita;

B)Alteração

- O ator seleciona um conceito já cadastrado com um duplo clique na tela "Concept Creation";
- O sistema habilita os botões "Alter" e "Delete" e todas as informações sobre o conceito são copiadas para os seus respectivos campos de criação;
- Quando o usuário clica no botão "Alter" o sistema verifica se o nome do conceito é único no projeto;
- Se o nome for único o sistema atualiza a tela "Concept Creation"
- Se não for único o sistema exibe uma tela pedindo ao usuário para inserir um novo nome;
- Se ator clicar no botão "Cancel" a operação de alteração é abortada e nenhuma modificação é feita;

5.4.3 Requerimentos Especiais

Não aplicável.

5.4.4 Pós-Condições

Não se aplica.

5.4.5 Pontos de Extensão

Não Há.

6 Adicionar, Alterar, Excluir Cenário – GMT005

6.1 Breve Descrição

Destina-se ao cadastro de novos cenários de testes.

6.2 Atores

Este caso de uso é de uso de todos os usuários do sistema.

6.3 Pré-Condições

Para que se possa cadastrar um conceito é necessário haver um projeto corretamente cadastrado.

6.4 Fluxo de Eventos

Não se aplica.

6.4.1 Fluxo Básico

- O sistema mostra a primeira tela do wizard de criação de cenários com os botões "Alter", "Delete" e "View Parameters" desabilitados e com o campo:
  - ✓ Scenary Name
- Quando o usuário clicar no botão "Add Scenary" o sistema verifica se o nome do cenário é único no projeto;

- Se o nome for único o sistema grava os dados, atualiza a arvore do projeto e mostra a segunda tela do Wizard que é a tela de cadastro de Parametro;
- Se não for único o sistema exibe uma tela pedindo ao usuário para inserir um novo nome;

A tela de cadastro de Cenário é a seguinte:

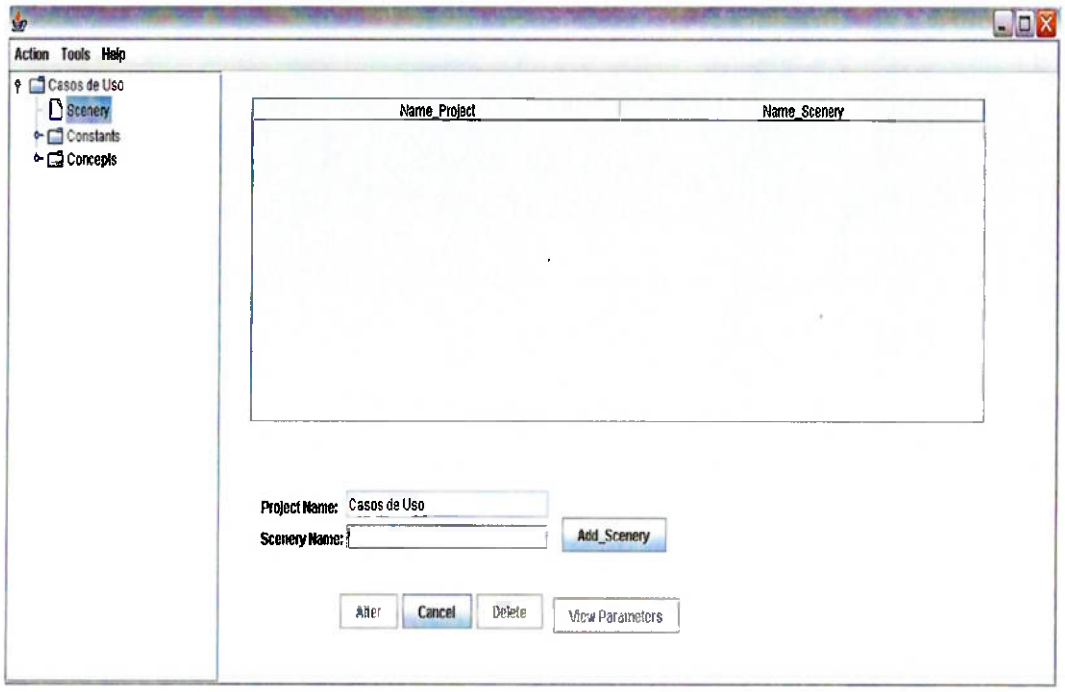

Figura 6.1 Tela de cadastro de Cenário(Add Scenery)

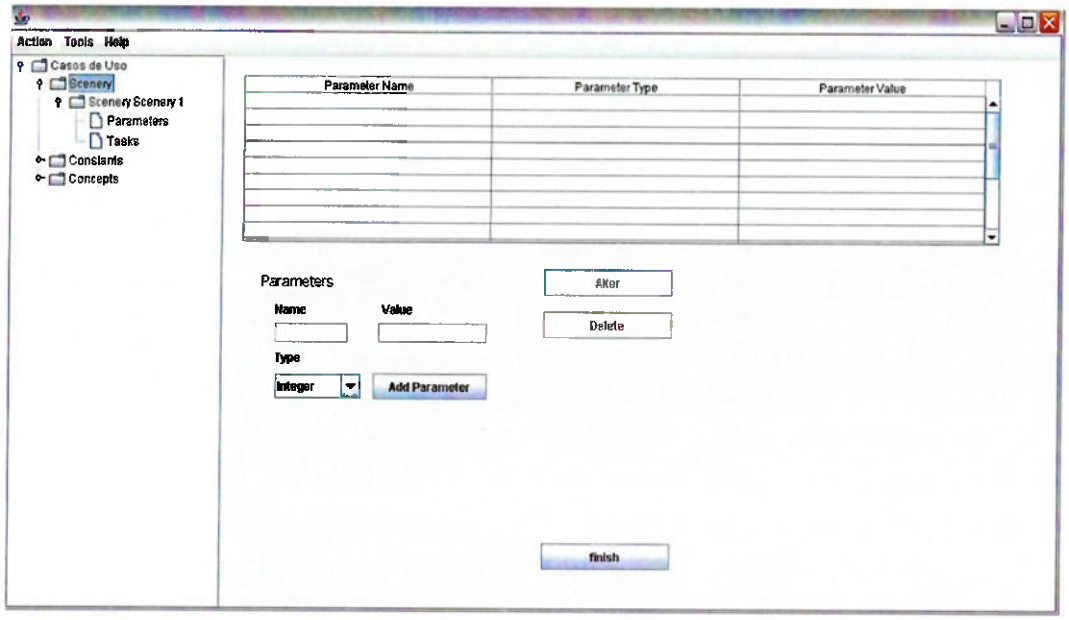

Figura 6.2 Tela após o cadastro de um cenário

6.4.2 Fluxos Alternativos

A)Exclusão de um Cenario

- O ator seleciona um cenario já cadastrado na tela "Add Scenary";
- O Sistema habilita os botões "Alter", "Delete" e "View Parameters"
- O Sistema copia o nome do cenário para o campo "Name Scenery";
- Se o ator clicar no botão "Delete" o sistema exibe uma mensagem padrão pedindo a confirmação da exclusão;
- Se o ator confirmar a exclusão a tela "Add Scenary" é atualizada e os dados são gravados;
- Se o ator não confirmar a exclusão o processo é abortado e nenhuma modificação é feita;

B)Alteração

- O ator seleciona um cenário já cadastrado na tela "Add Scenary";
- O Sistema habilita os botões "Alter", "Delete" e "View Parameters"
- O Sistema copia o nome do cenário para o campo "Name Scenery";
- Quando o usuário clicar no botão "Alter" o sistema verifica
- se o nome é único no projeto
- Se for unico o sistema atualiza a tela de cadastro de Cenário, grava os dados e abra e a tela seguinte do wizard(tela de cadastro de Parâmetro)
- Se não for único o sistema exibe uma tela pedindo ao usuário para inserir um novo nome;

C)Mostrar Parâmetros

- O ator seleciona um cenário já cadastrado na tela "Add Scenary";
- O Sistema habilita os botões "Alter", "Delete" e "Show Parameter"
- O Sistema copia o nome do cenário para o campo "Name Scenery";
- Quando o usuário clicar no botão "View Parameters" o sistema abre a tela de cadastro de Parâmetros

Se ator clicar no botão "Cancel" a operação de alteração é abortada e nenhuma modificação é feita;

6.4.3 Requerimentos Especiais

Não aplicável.

6.4.4 Pós-Condições

Após o cadastramento do Cenário deve-se abrir a segunda tela do wizard, relativa a criação de parâmetros, descrita no próximo caso de uso.

6.4.5 Pontos de Extensão

Não se aplica.

7 Adicionar, Alterar e Excluir Parâmetro - GMT006

7.1 Breve Descrição

Destina-se a cadastrar parâmetros aos cenários criados na tela Wizard "Add Scenary";

7.2 Atores

Este caso de uso é de uso de todos os usuários do sistema.

7.3 Pré-Condições

É necessário ter um cenário previamente cadastrado.

7.4 Fluxo de Eventos

Não se aplica.

7.4.1 Fluxo Básico

- O sistema mostra a tela de criação de parâmetros com os botões "Alter" e "Delete" desabilitados e com os campos:
  - ✓ Name
  - ✓ Type(Esse campo só pode assumir os valores Integer, Double e String)
  - ✓ Values(Os valore devem ser inseridos com espaços e sem virgulas)

- Quando o usuário clicar no botão "Add Parameter" o sistema verifica se o nome do parâmetro é único no Cenario e se os valores fornecidos são condizentes com o tipo selecionado;
- Se o nome for único o sistema grava os dados e atualiza a arvore do projeto e a tela "Add Parameter";
- Se não for único o sistema exibe uma tela pedindo ao usuário para inserir um novo nome;
- Quando o usuário clicar no botão "Finish" a tela é encerrada voltando para a tela "Add Scenery";

A tela de cadastro de parâmetro("Add Parameter") é a seguinte

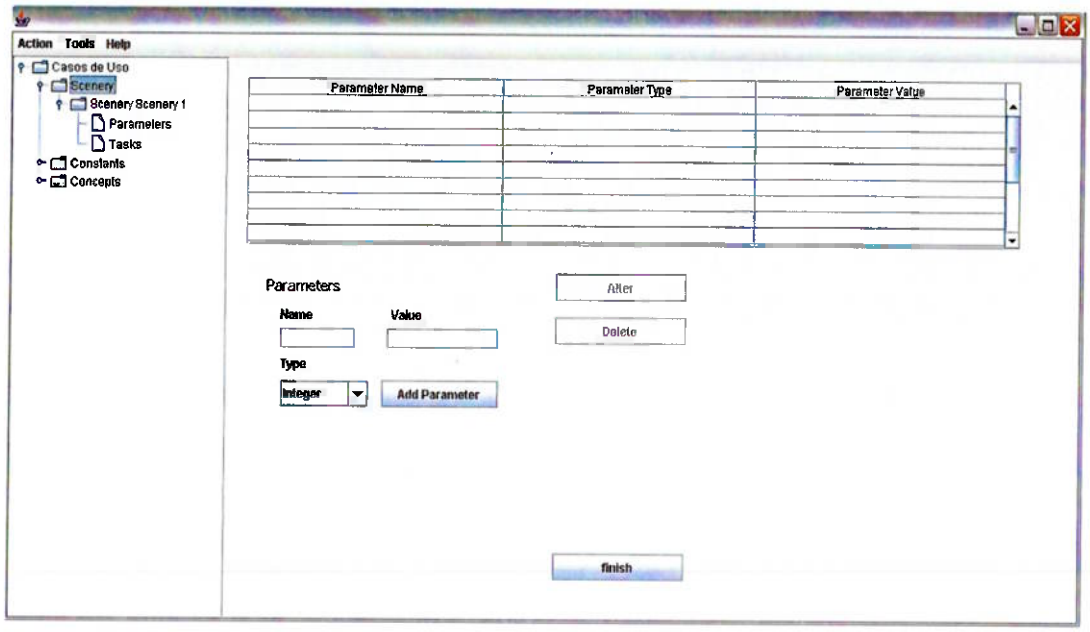

Figura 7.1 Tela de cadastro de Parâmetro(Add Parameter)

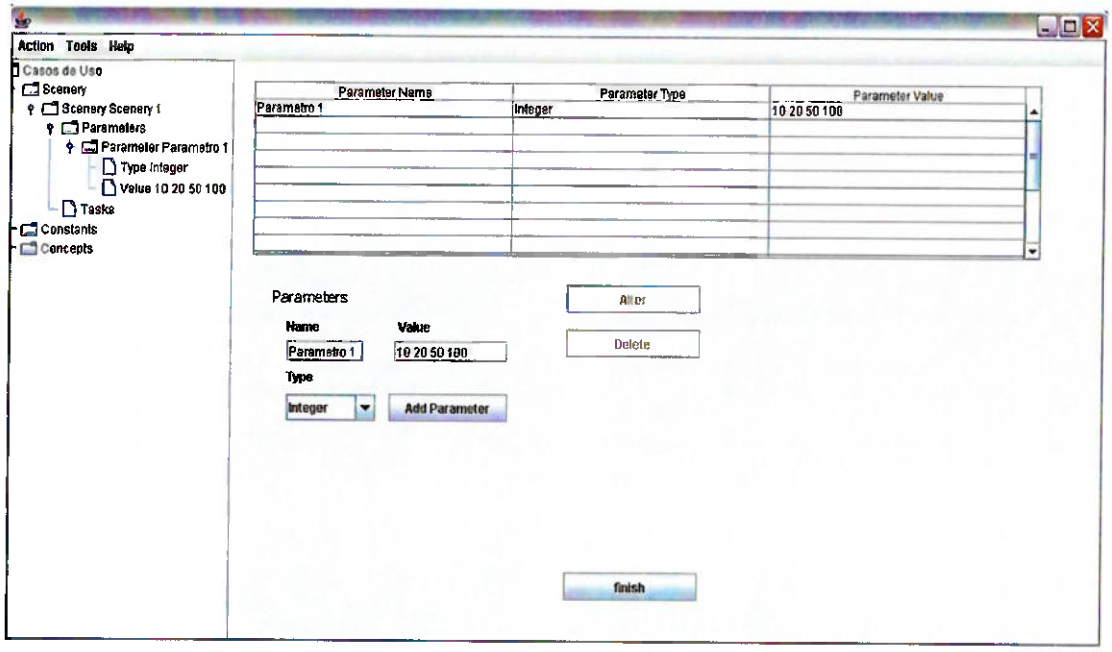

Figura 7.2 Tela após o cadastro de um Parâmetro

7.4.2 Fluxos Alternativos

A)Exclusão de um parâmetro

- O ator seleciona um parâmetro já cadastrado na tela "Add Parameter";
- O sistema habilita os botões "Alter" e "Delete" e copia os dados do parâmetro selecionado para os seus respectivos campos;
- Se o ator clicar no botão "Delete" o sistema exibe uma mensagem padrão pedindo a confirmação da exclusão;
- Se o ator confirmar a exclusão a tela "Add Parameter" é atualizada e os dados são gravados;

- Se o ator não confirmar a exclusão o processo é abortado e nenhuma modificação é feita;

B)Alteração

- O ator seleciona um cenário já cadastrado na tela "Add Parameter";
- O sistema habilita os botões "Alter" e "Delete" e copia os dados do parâmetro selecionado para os seus respectivos campos;
- Quando o usuário clicar no botão "Accept Alter" o sistema verifica se o nome é único no cenário;
- Se for único o sistema atualiza a arvore do projeto tela "Add Parameter" e grava os dados
- Se não for único o sistema exibe uma tela pedindo ao usuário para inserir um novo nome;

7.4.3 Requerimentos Especiais

Não aplicável.

7.4.4 Pós-Condições

Não aplicável.

7.4.5 Pontos de Extensão

Não aplicável.

8 Adicionar, Alterar, Excluir Caso – GMT008

8.1 Breve Descrição

Destina-se ao cadastro de casos aos cenários já criados.

8.2 Atores

Este caso de uso é de uso de todos os usuários do sistema

8.3 Pré-Condições

O ator deve ter criado um cenário para poder adicionar casos a ele.

8.4 Fluxo de Eventos

Não se aplica.

8.4.1 Fluxo Básico

- O sistema mostra a tela de criação de casos com os botões "Alter" e "Delete" desabilitado e os campos:
  - ✓ Concept(Check Box com todos os Conceitos cadastrados)
  - ✓ Operation(Pode assumir os valores =, <, >, !=)
  - ✓ Value
  - ✓ Parameter(Check Box com todos os Parâmetros cadastrados naquele Cenário)

- ✓ Constant(Check Box com todos as Constantes cadastrados no projeto)
- ✓ Unic
- ✓ Multiplayer
- ✓ Case Name
- ✓ Rule

- Quando o usuário clicar no botão "Mount Rule" o sistema escreve no campo Rule uma regra conforme os itens selecionados pelo usuário da seguinte forma:
    1. Copia-se para uma String o nome do conceito selecionado com um espaço em branco no final do nome;
    2. Soma-se a String a operação selecionada com um espaço em branco no final;
    3. Verifica-se qual tipo de valor foi selecionado(Value, Constant ou Parameter)
        - Se for Value soma-se a string um v minúsculo com um espaço em branco e soma-se o valor do campo value;
        - Se for Constant soma-se a string um c minúsculo com um espaço em branco e soma-se o nome da constante selecionada;
        - Se for Parameter soma-se a string um p minúsculo com um espaço em branco e soma-se o nome do parâmetro selecionado;
    4. Soma-se a String resultante a String que já contiver no campo Rule
    5. Se quiser adicionar mais uma regra ao mesmo caso deve-se clicar no botão AND ou OR para que seja adicionado ao final do campo Rule um espaço em

branco, mais a palavra AND ou OR e outro espaço em branco.

- Quando o usuário clicar no botão "Add task" o sistema verifica se o nome do caso é único no cenário;
- Se o nome for único o sistema grava os dados e atualiza a tela "Add Task";
- Se não for único o sistema exibe uma tela pedindo ao usuário para inserir um novo nome;
- Quando o usuário clicar no botão "Cancel" a tela é atualizada;

A tela será a seguinte:

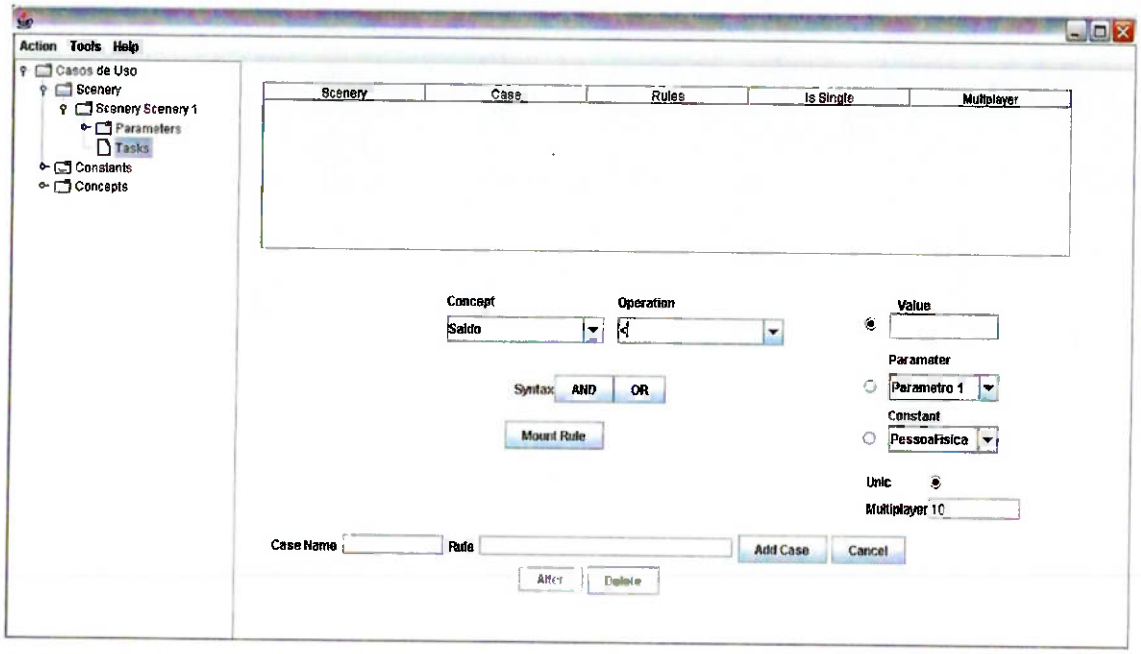

Figura 8.1 Tela de cadastro de Caso(Add Task)

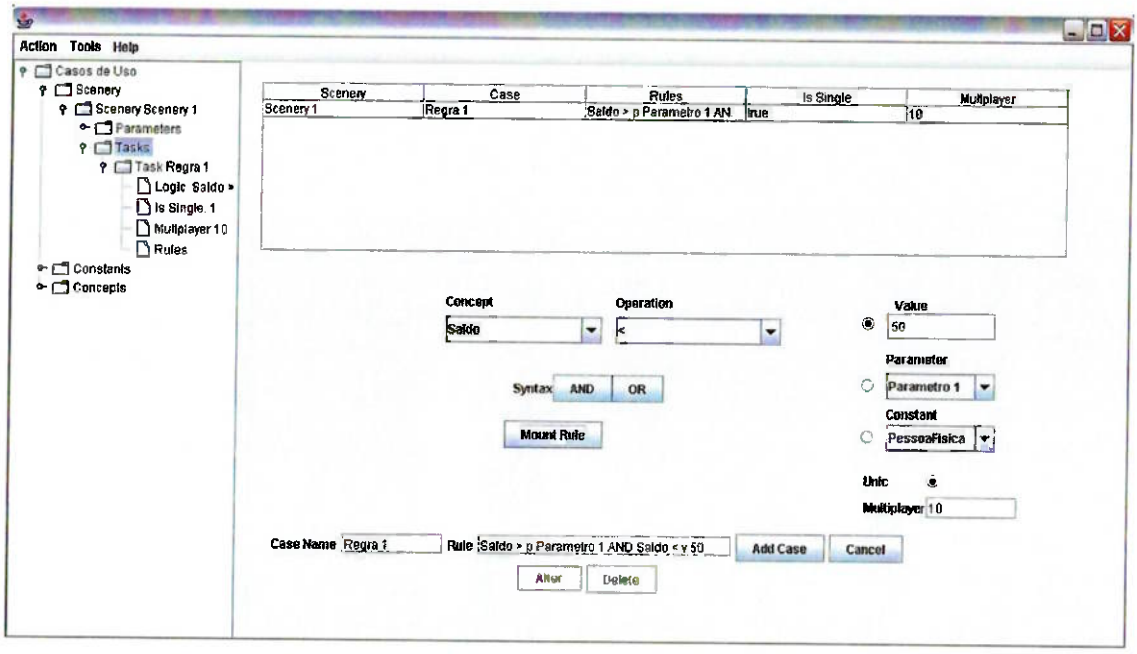

Figura 8.2 Tela após o cadastramento de um Caso

8.4.2 Fluxos Alternativos

A)Exclusão de um Parâmetro

- O ator seleciona um na tabela de Caso
- O sistema habilita os botões "Alter" e "Delete" e todas as informações sobre o Caso são copiados para os seus respectivos campos de criação;
- Se o ator clicar no botão "Delete" o sistema exibe uma mensagem padrão pedindo a confirmação da exclusão;
- Se o ator confirmar a exclusão a tela "Add Task" é atualizada e os dados são gravados;
- Se o ator não confirmar a exclusão o processo é abortado e nenhuma modificação é feita;

B)Alteração

- O ator seleciona um cenário já cadastrado na tela "Add Task";
- O sistema habilita os botões "Alter" e "Delete" e todas as informações sobre o Caso são copiados para os seus respectivos campos de criação;
- Quando o usuário clicar no botão "Alter" o sistema verifica se o nome é único no cenário;
- Se for único o sistema atualiza a arvore do Projeto e a tela "Add Task" e grava os dados
- Se não for único o sistema exibe uma tela pedindo ao usuário para inserir um novo nome;

8.4.3 Requerimentos Especiais

Não aplicável.

8.4.4 Pós-Condições

Não se aplica.

8.4.5 Pontos de Extensão

Não se aplica.

9 Adicionar, Alterar e Excluir Regra – GMT009

9.1 Breve Descrição

Destina-se a adicionar novas regras aos casos já criados.

9.2 Atores

Este caso de uso é de uso de todos os usuários do sistema.

9.3 Pré-Condições

O ator deve ter criado um caso para poder adicionar regras a ele.

9.4 Fluxo de Eventos

Não se aplica.

9.4.1 Fluxo Básico

- O sistema mostra a tela de criação de regras com os botões "Alter" e "Delete" desabilitado e com os campos:
  - ✓ Concept(Check Box com todos os Conceitos cadastrados)
  - ✓ Operation(Pode assumir os valores =, <, >, !=)
  - ✓ Value
  - ✓ Parameter(Check Box com todos os Parâmetros cadastrados naquele Cenário)
  - ✓ Constant(Check Box com todos as Constantes cadastrados no projeto)

- ✓ Unic
- ✓ Multiplayer
- ✓ Case Name
- ✓ Rule
- ✓ Output

- Quando o usuário clicar no botão "Mount Rule" o sistema escreve no campo Rule uma regra conforme os itens selecionados pelo usuário da seguinte forma:
    1. Copia-se para uma String o nome do conceito selecionado com um espaço em branco no final do nome;
    2. Soma-se a String a operação selecionada com um espaço em branco no final;
    3. Verifica-se qual tipo de valor foi selecionado(Value, Constant ou Parameter)
        - Se for Value soma-se a string um v minúsculo com um espaço em branco e soma-se o valor do campo value;
        - Se for Constant soma-se a string um c minúsculo com um espaço em branco e soma-se o nome da constante selecionada;
        - Se for Parameter soma-se a string um p minúsculo com um espaço em branco e soma-se o nome do parâmetro selecionado;
    4. Soma-se a String resultante a String que já contiver no campo Rule
    5. Se quiser adicionar mais uma regra ao mesmo caso deve-se clicar no botão AND ou OR para que seja adicionado ao final do campo Rule um espaço em

branco, mais a palavra AND ou OR e outro espaço em branco.

- Quando o usuário clicar no botão "Add Rule" o sistema verifica se o nome da regra é único no caso;
- Se o nome for único o sistema grava os dados e atualiza a arvore do projeto e a tela "Add Rule";
- Se não for único o sistema exibe uma tela pedindo ao usuário para inserir um novo nome;
- Quando o usuário clicar no botão "Cancel" a tela é encerrada;

A tela será a seguinte:

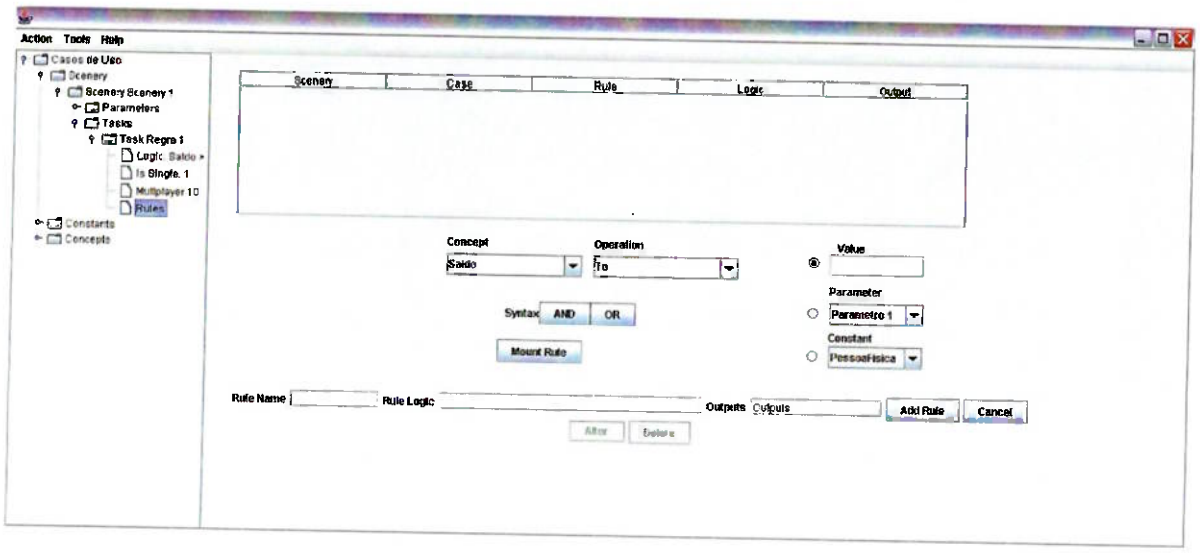

Figura 9.1 Tela de cadastro de Regra(Add Rule)

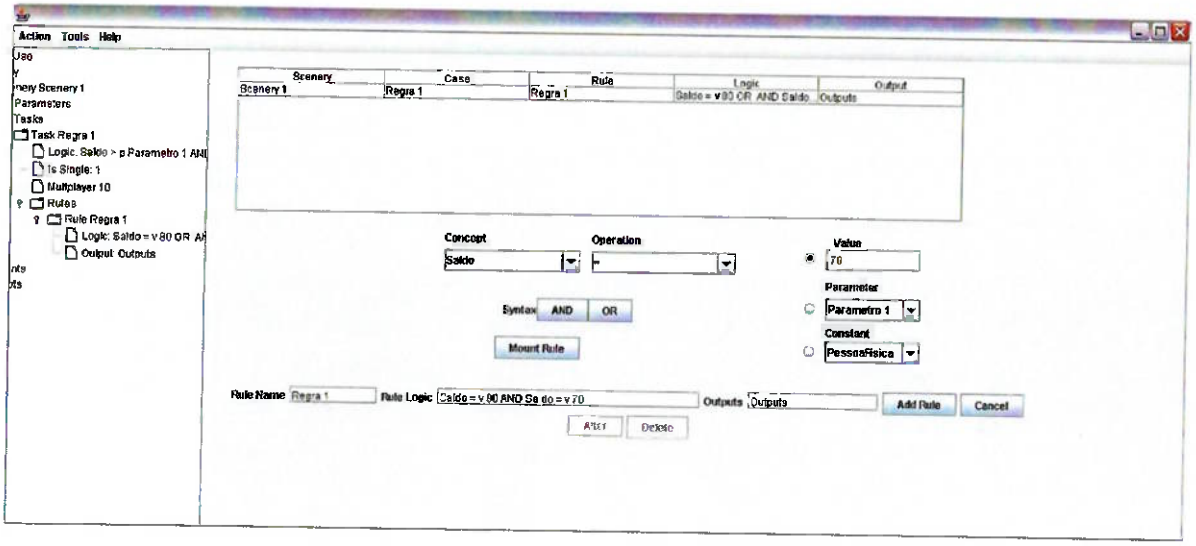

Figura 9.2 Tela após o cadastro de uma regra

9.4.2 Fluxos Alternativos

A)Exclusão de uma Regra

- O ator seleciona uma regra já cadastrada na tabela de Regras
- O sistema habilita os botões "Alter" e "Delete" e todas as informações sobre a regra selecionada são copiadas para os seus respectivos campos de criação;
- Se o ator clicar no botão "Delete" o sistema exibe uma mensagem padrão pedindo a confirmação da exclusão;
- Se o ator confirmar a exclusão a tela "Add Rule" é atualizada e os dados são gravados;
- Se o ator não confirmar a exclusão o processo é abortado e nenhuma modificação é feita;

B)Alteração

- O ator seleciona uma regra já cadastrada na tela "Add Rule";

- O sistema habilita os botões "Alter" e "Delete" e todas as informações sobre a regra selecionada são copiadas para os seus respectivos campos de criação;
- Quando o usuário clicar no botão "Accept Alter" o sistema verifica se o nome da regra é único no cenário;
- Se for único o sistema atualiza a tela "Add Rule" e grava os dados
- Se não for único o sistema exibe uma tela pedindo ao usuário para inserir um novo nome;

9.4.3 Requerimentos Especiais

Não aplicável.

9.4.4 Pós-Condições

Não aplicável.

9.4.5 Pontos de Extensão

Nenhum.

10 Gerar Massa de Testes e Roteiro – GMT012

10.1 Breve Descrição

Destina-se a seleção dos cenários escolhidos, do caminho de gravação do arquivo txt e confirmação da geração.

10.2 Atores

Este caso de uso é de uso de todos os usuários do sistema.

10.3 Pré-Condições

Deve existir ao menos uma regra cadastrada.

10.4 Fluxo de Eventos

Não se aplica.

10.4.1 Fluxo Básico

- O sistema mostra a tela "New Generate" com o campo:
  - ✓ Path
- Quando o usuário clica no botão botão "Generate Mass of Test" .o sistema verifica quais cenários estão selecionados e verifica quais registros são relevantes para cada cenário gravando a resposta em um arquivo txt no formato CVS no diretório que está o programa

A tela é a seguinte:

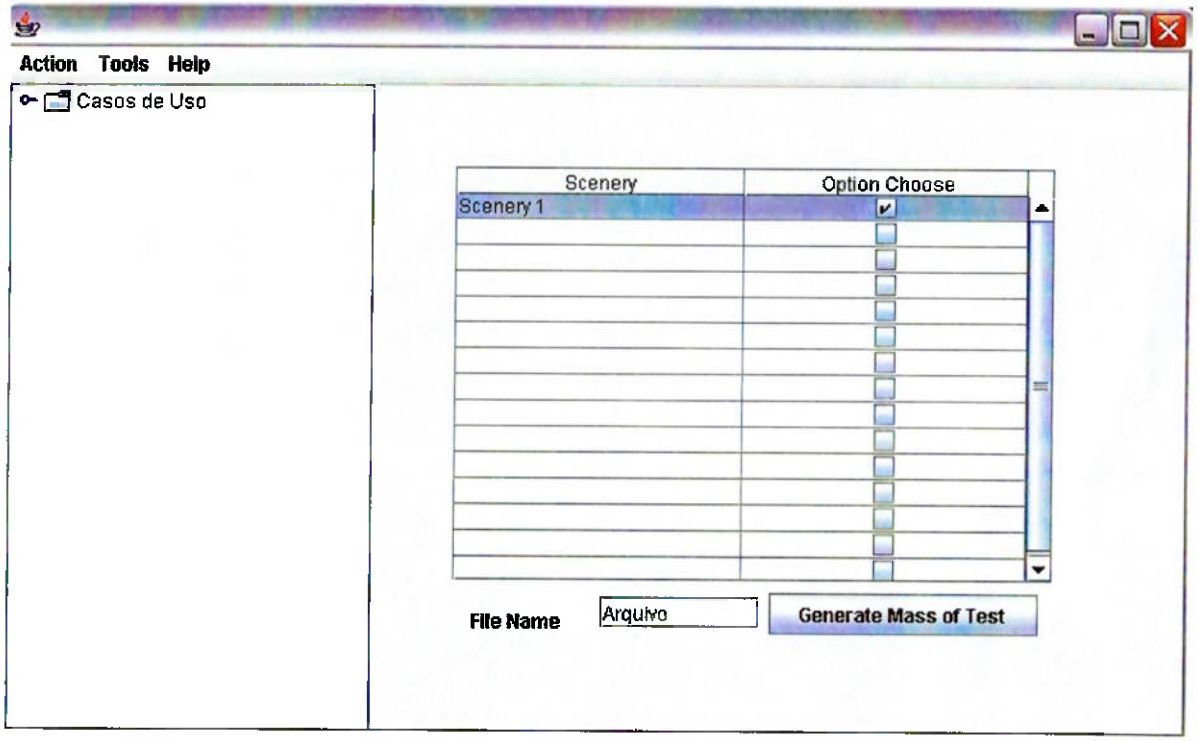

Figura 10.1 Tela de Geraçao da Base

10.4.2 Fluxos Alternativos

Não Aplicável.

10.4.3 Requerimentos Especiais

Este é o core do programa. A primeira parte é composta por funções responsáveis pela leitura e quebra das Strings de regras e casos, gerando informações que comporão a consulta. Assim, a regra é quebrada em tabelas, conceitos, valores operações e operadores lógicos sendo gravada em vetores de Strings.

Em seguida, todas as tabelas são reclassificadas no vetor por ordem alfabética e as funções de montagem são chamadas. Essas funções remontam as regras na forma de chamadas sql. Para tanto conectam-se as regras de casos e suas regras personalizadoras através da variável lógica "and".

Com a chamada montada, procedemos com a consulta e os registros encontrados, advindos das regras cadastradas, são gravados em um arquivo txt, que será disponibilizado para o usuário.

Em resumo, o programa tem duas utilidades. A primeira delas é a de orientação na formulação dos cenários de testes. Procura-se automatizar e padronizar o processo, fazendo com que o usuário pense e estude quais cenários são relevantes à sua pesquisa e qual será a estratégia de testes.

Além disso, através de cadastros simples, montam-se consultas complexas na base de dadso, facilitando e garantindo que a massa de testes gerada realmente é adequada para aquele determinado caso.

10.4.4 Pós-Condições

Deve ser gerado um roteiro de testes e uma base de dados relevantes para este roteiro.

10.4.5 Pontos de Extensão

Nenhum.