Ferramenta de Suporte ao Teste Funcional de Software a Partir de Diagramas de Casos de Uso

Acadêmico: Juliano Bianchini

Orientador: Everaldo Artur Grahl

FURB/BCC

Disciplina de Trabalho de Conclusão de Curso II

Julho de 2004

Roteiro da Apresentação

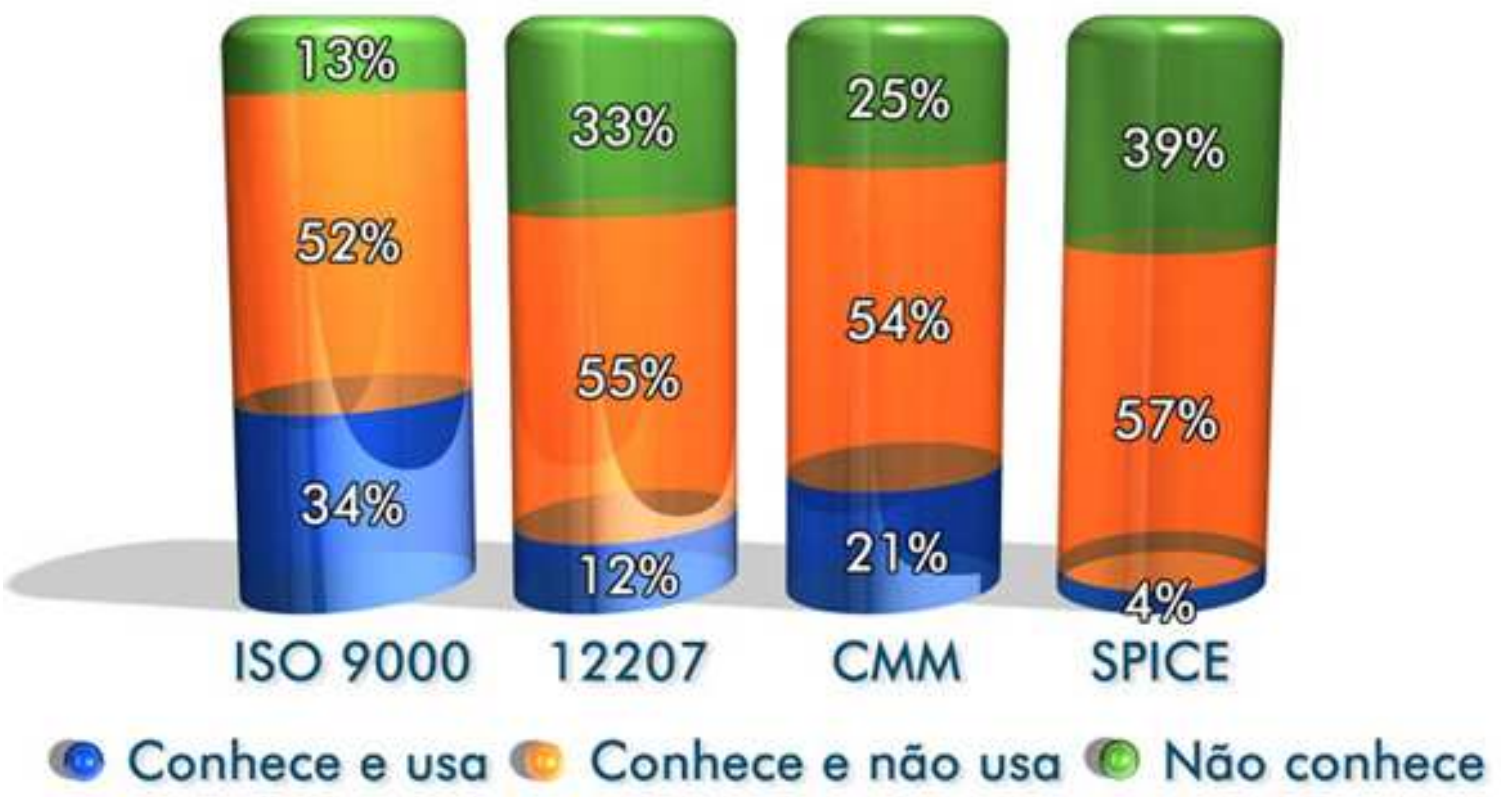
- **Introdução**
- **Objetivos do trabalho**
- **Fundamentação teórica**
  - Qualidade e teste de software;
  - Utilização de casos de uso no teste funcional de software;
  - Padrão IEEE 829-1998;
  - Ferramenta CASE para modelagem UML – ArgoUML.
- **Desenvolvimento do trabalho**
  - Requisitos principais do problema;
  - Especificação;
  - Implementação;
  - Técnicas e ferramentas utilizadas;
  - Operacionalidade da implementação;
  - Resultados e discussões;
  - Conclusões e extensões.

Introdução

- Grande parte dos problemas de qualidade encontrados em software pode ser atribuído a (RYSER; GLINZ, 2003, p. 1-2):
  - Falta de planejamento dos tempos e custos;
  - Falta de documentação;
  - O teste é a última etapa do processo de desenvolvimento;
  - Os casos de teste não são gerados de forma sistemática;

Introdução

- No Brasil, fala-se muito e faz-se pouco...



Fonte: Secretaria de Política de Informática (SEPIN, 2004)

Introdução

- Utilização da UML:
  - Para modelar sistemas orientados a objetos;
  - Como modelo sistemático para o desenvolvimento dos teste;

- O diagrama de caso de uso tem sido estudado como modelo para o teste funcional;
  - Propostas: Heumann (2004) e Binder (2000, p. 722-731);

Objetivos do Trabalho

- Análise e aplicação das metodologias de teste baseadas em casos de uso da UML apresentadas por Heumann (2004) e Binder (2000, p. 722-731);
- Adaptação de uma ferramenta CASE de código aberto para suportar extensões aplicáveis a casos de testes funcionais;
- Incorporação de padrões de documentação de teste de software da IEEE 829 na ferramenta de apoio a execução e documentação do processo de teste funcional que foi construída;

Qualidade de Software

- Qualidade de software: focaliza todas as etapas e artefatos produzidos com o objetivo de garantir a conformidade de processos e produtos, prevenindo e eliminando defeitos (BARTIÉ, 2002, p. 16);
- Verificação e Validação: assegura que o software cumpra com suas especificações e atenda às necessidades dos clientes (SOMMERVILLE, 2003, p. 358);
- Validação: estamos construindo o software certo?
- Verificação: estamos construindo certo o software?

Teste de Software

- Teste: um processo sistemático e planejado que tem por finalidade a identificação de erros (BARTIÉ, 2002, p. 22);
- Teste é uma das técnicas de V&V;
- Teste é uma tarefa destrutiva...
- Formas de teste:
  - Caixa preta: verifica se as funcionalidades do software são realizadas adequadamente;
  - Caixa branca: verifica a estrutura interna do software, testando os caminhos lógicos.

Teste de Software

- Estágios:
  - Unidade: testa-se a menor unidade de software. É orientado a caixa branca;
  - Integração: as unidades são integradas. Também é orientado a caixa branca;
  - Sistema: testa-se todo o sistema, levando em consideração componentes de software e hardware;
  - Validação ou aceitação: é feito por intermédio de uma série de testes de caixa preta que demonstram conformidade com os requisitos.

Casos de Uso como Modelo para o Teste

- Diagramas de caso de uso descrevem requisitos e representam a forma como o sistema será usado;
- Mostram ao usuário o que esperar do software, ao desenvolverdor o que codificar, ao "documentador" o que documentar e ao testador o que testar (HEUMANN, 2004);

- Podem ajudar na verificação e validação:
  - Verificando se o os requisitos foram implementados;
  - Testando e validando a implementação dos requisitos.

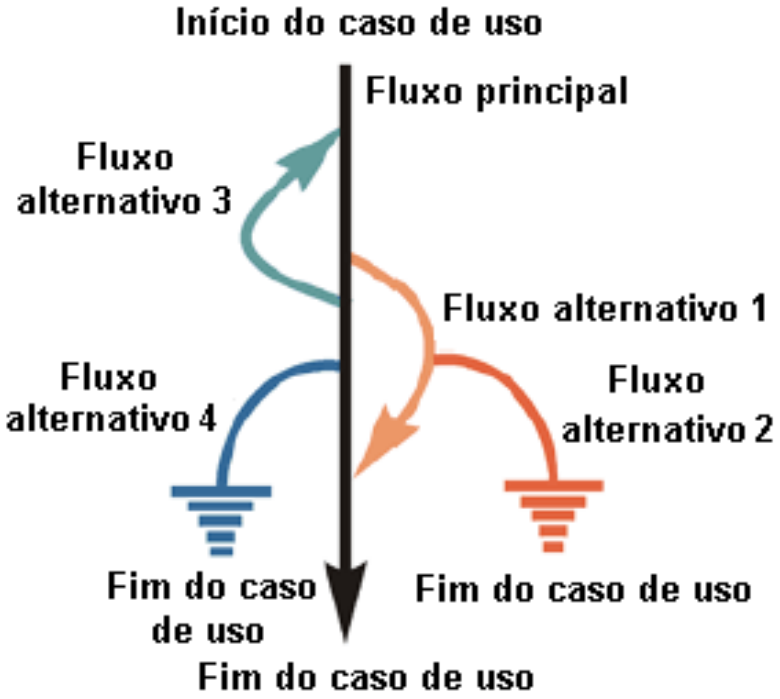
Casos de Uso como Modelo para o Teste

- Simplificam o processo de teste, incrementam a eficiência e ajudam na certeza de se ter uma cobertura completa dos testes (HEUMANN, 2004);

- Propostas: Heumann (2004) e Binder (2000, p. 722-731);

Teste baseado em Casos de Uso (HEUMANN, 2004)

- 1) Para cada caso de uso, gerar uma lista de cenários de casos de uso:
  - Ler a descrição textual do caso de uso e identificar o fluxo principal e os fluxos alternativos;



Teste baseado em Casos de Uso (HEUMANN, 2004)

- 2) Para cada cenário identificar, ao menos, um caso de teste e as condições que o farão ser executado:
  - Analisar os cenários, revisar a descrição do caso de uso e criar o caso de teste;
  - Elaborar uma tabela que represente valores de entrada e saída pode ajudar na organização e montagem dos casos de teste;
- 3) Para cada caso de teste, identificar os dados que serão utilizados no teste:
  - Identificar os dados e revisar cada caso para assegurar a meticulosidade e identificar redundâncias ou faltas.

Teste de Caso de Uso Estendido (BINDER, 2000)

- 1) Identificar as variáveis operacionais:
  - Variáveis que são explicitamente parte da interface que suporta o caso de uso tais como entradas e saídas do sistema;
- 2) Identificar os domínios das variáveis operacionais:
  - Definição de quais são os valores válidos e inválidos para cada variável;

Teste de Caso de Uso Estendido (BINDER, 2000)

- 3) Desenvolver os relacionamentos operacionais:
  - Modelagem do relacionamento entre as variáveis operacionais que determinam diferentes respostas do sistema.
  - Pode-se modelar os relacionamentos através de uma tabela de decisão;
- Ex.:

Variante		Variáveis operacionais		Resultados esperados
		username	senha	
1	Válida	fulano	xpto123	O sistema é aberto na tela principal
2	Inválida	fulano	xyz321	Mensagem de username/senha inválido

Teste de Caso de Uso Estendido (BINDER, 2000)

- 4) Desenvolver casos de teste:
  - Escrever os casos de teste com base no relacionamento entre as variáveis operacionais;
  - Os resultados esperados podem ser desenvolvidos pela observação dos valores de entrada.

Teste de Caso de Uso Estendido (BINDER, 2000)

- Vantagens:
  - A utilização dos casos de uso está consolidada nos processos de análise e projeto, facilitando o desenvolvimento dos casos de teste;
  - Casos de uso refletem o ponto de vista do usuário;
  - Provêem uma forma sistemática de desenvolvimento das informações necessárias para o projeto de teste;
  - Casos de uso ambíguos, inconsistentes ou incompletos, são logo apontados pelos testes.

Teste de Caso de Uso Estendido (BINDER, 2000)

- Desvantagens:
  - Casos de uso não são usados para especificar, entre outras, performance e tolerâncias a falhas;
  - Esta metodologia não está preparada para suportar as dependências entre casos de uso (construtores do tipo *extends* e *includes*).

IEEE 829 – Padrão para Documentação do Teste de Software

- Tem por objetivo descrever os documentos necessários para apoiar a atividade de teste de software;

- Os documentos descritos neste padrão abrangem o planejamento, especificação e a geração de relatórios de testes;

IEEE 829 – Padrão para Documentação do Teste de Software

- É composto pelos documentos de:
  - Especificação do plano de teste: prescreve o escopo, o plano de ação, os recursos e o cronograma das atividades de teste;
  - Especificação do projeto de teste: refina o que foi definido no plano de teste e identifica as características que serão testadas;
  - Especificação do caso de teste: detalha cada característica que será testada;

IEEE 829 – Padrão para Documentação do Teste de Software

- É composto pelos documentos de (continuação):
  - Especificação do procedimento de teste: especifica os passos para execução do caso de teste;
  - Relatório de transição de item de teste: identifica os itens que estão sendo enviados para a equipe de testes;
  - *Log* dos testes (diário de bordo): provê um histórico cronológico dos detalhes relevantes da execução dos testes;

IEEE 829 – Padrão para Documentação do Teste de Software

- É composto pelos documentos de (continuação):
  - Relatório de incidente: documenta qualquer evento que ocorreu durante o processo de teste e que requer investigação;
  - Relatório com resumo dos testes: sumariza os resultados dos testes projetados.

A Ferramenta Case ArgoUML

- É uma ferramenta CASE (*Open Source*) para criação de diagramas da UML;
- Gera código-fonte em Java;
- Possibilita fazer a engenharia reversa a partir de código-fonte em Java;
- Utiliza diversos outros projetos de código-fonte aberto, tais como GEF (Graph Editing Framework), log4j, SAX Parser Factory e Novosoft UML Library;

Requisitos Principais do Problema

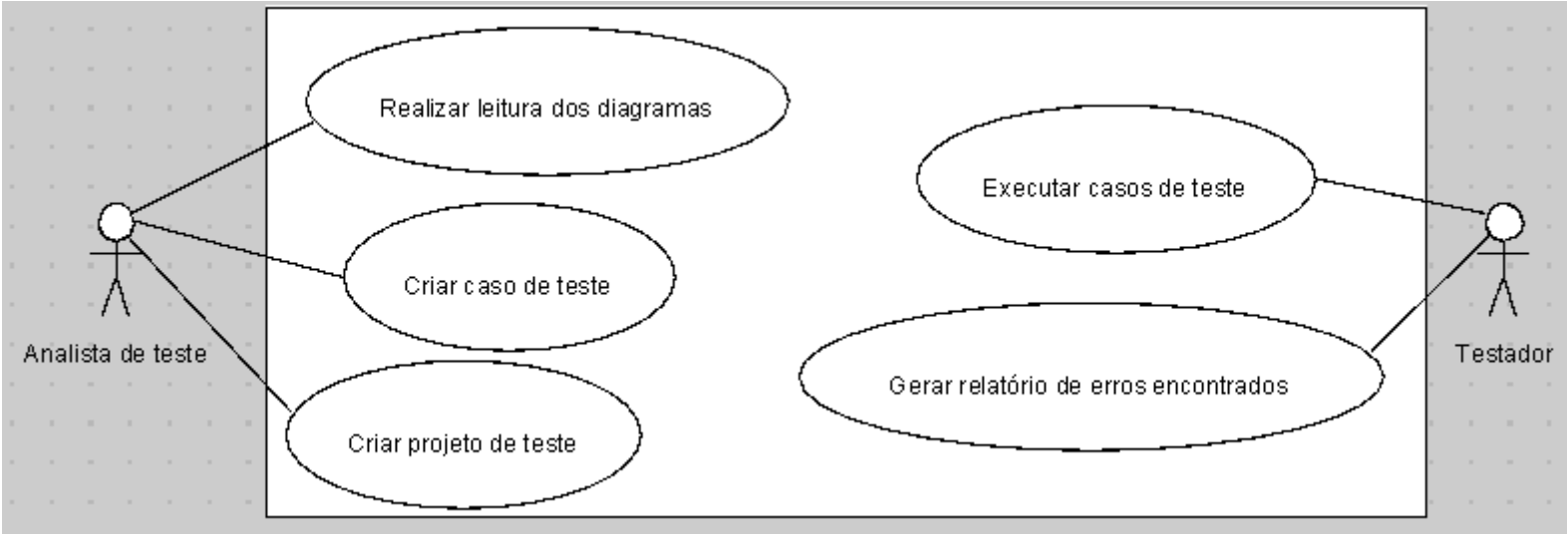
- Permitir a leitura dos casos de testes criados na ferramenta CASE ArgoUML;
- Permitir a manutenção dos cadastros necessários ao planejamento dos testes;
- Permitir o registro de informações sobre a execução dos casos de testes criados e armazenar o histórico destas execuções;

Requisitos Principais do Problema

- Emitir um relatório com os erros encontrados na execução de um caso de teste;
- Gerar gráficos sobre a cobertura de testes realizados e testes executados.

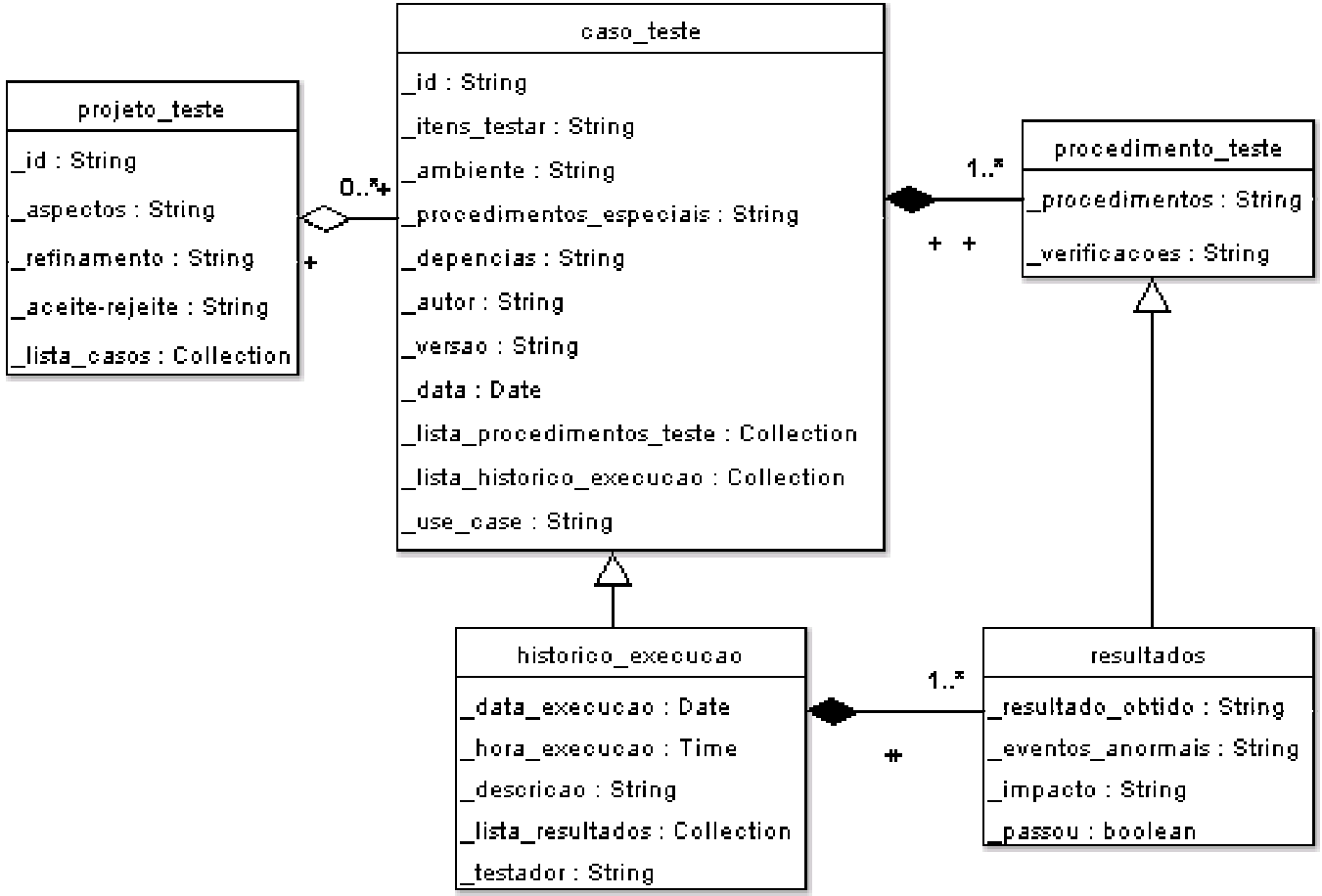
Especificação

- Digrama de casos de uso (principais):



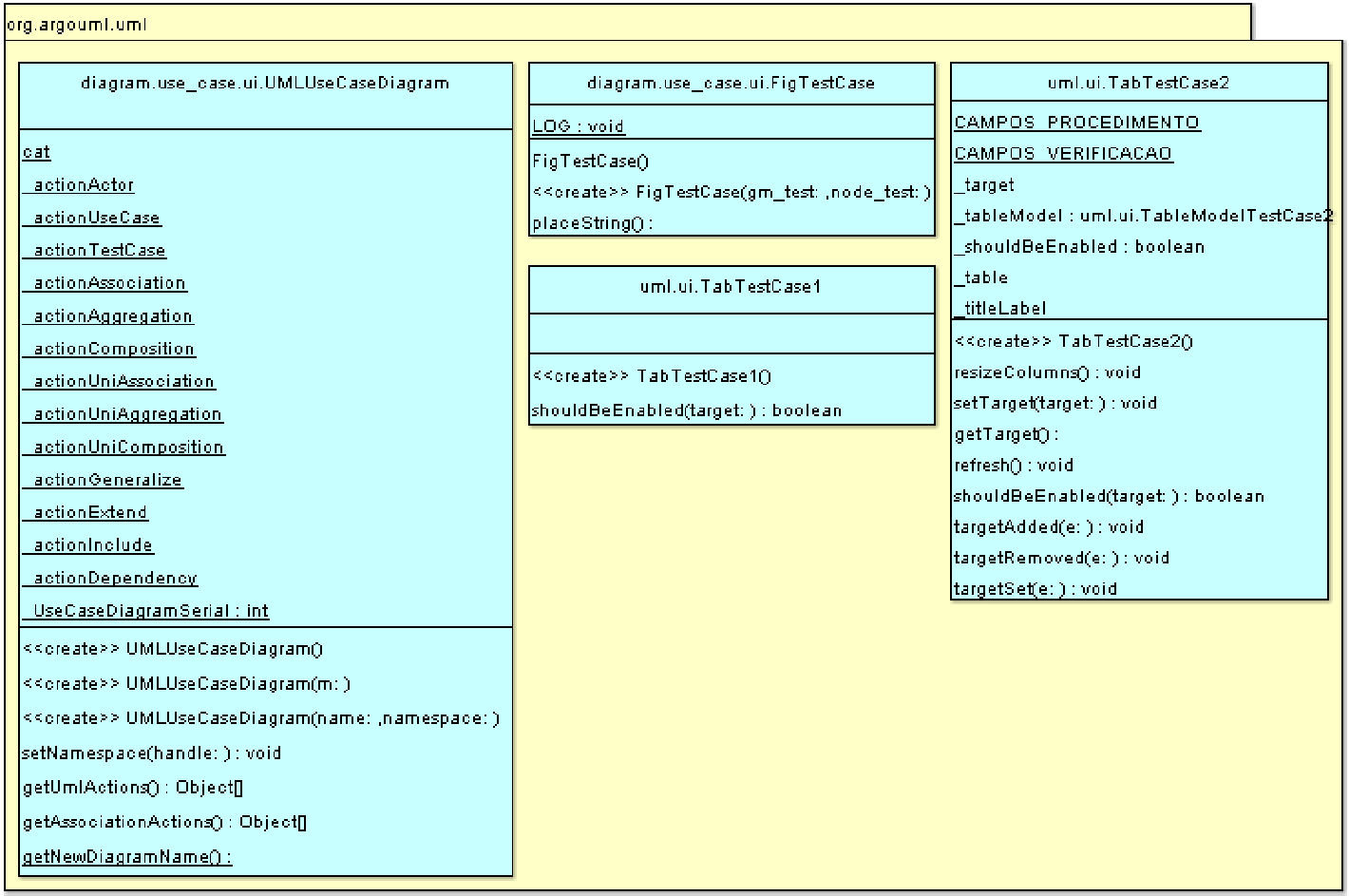
Especificação

- Digrama de classes (TestCen):



Especificação

- Digrama de classes (ArgoUML):

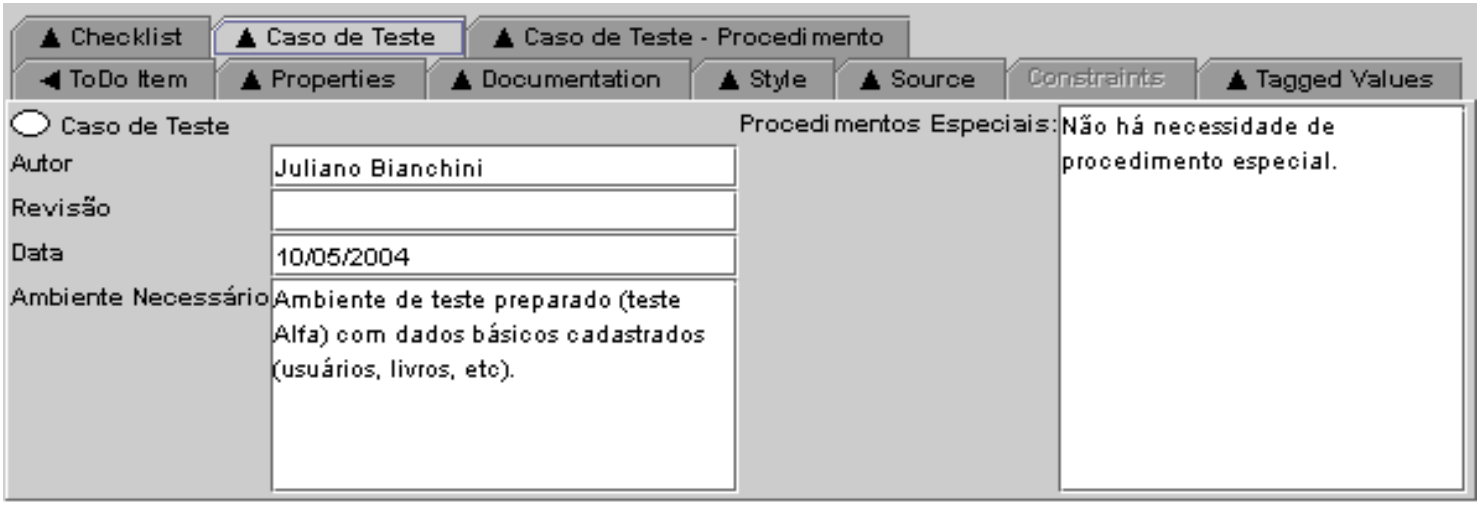


Implementação

- Alterações no ArgoUML:
  - Criação do estereótipo <<Caso de Teste>>;

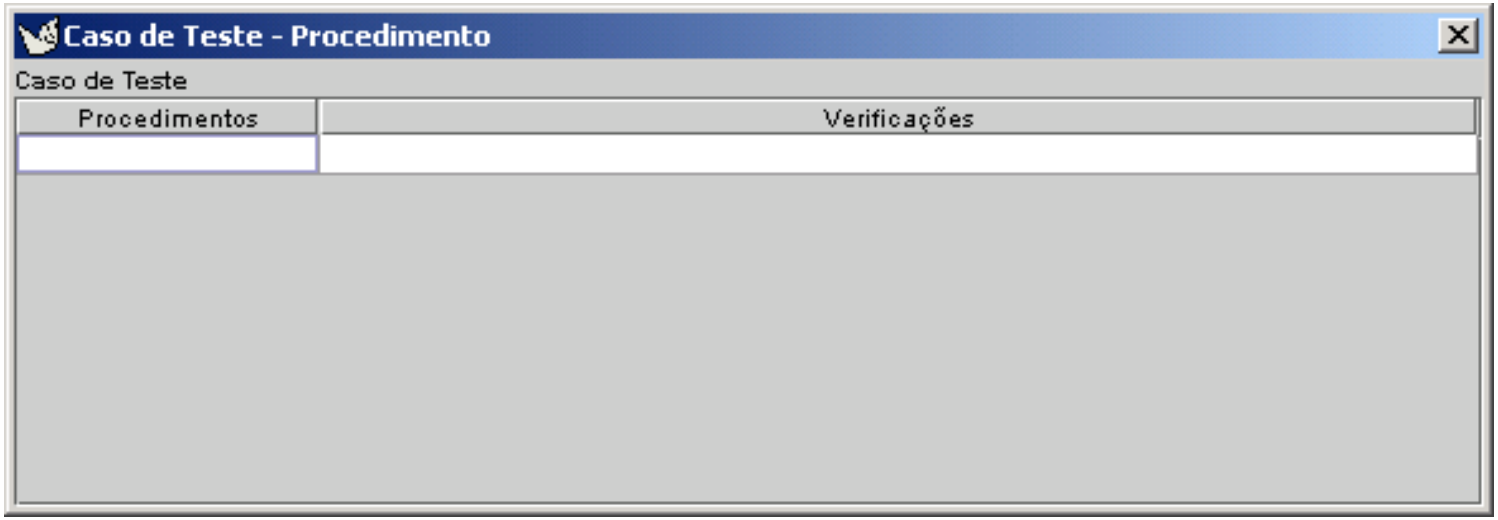
Implementação

- Alterações no ArgoUML:
  - Criação de duas guias para inserir informações para teste;



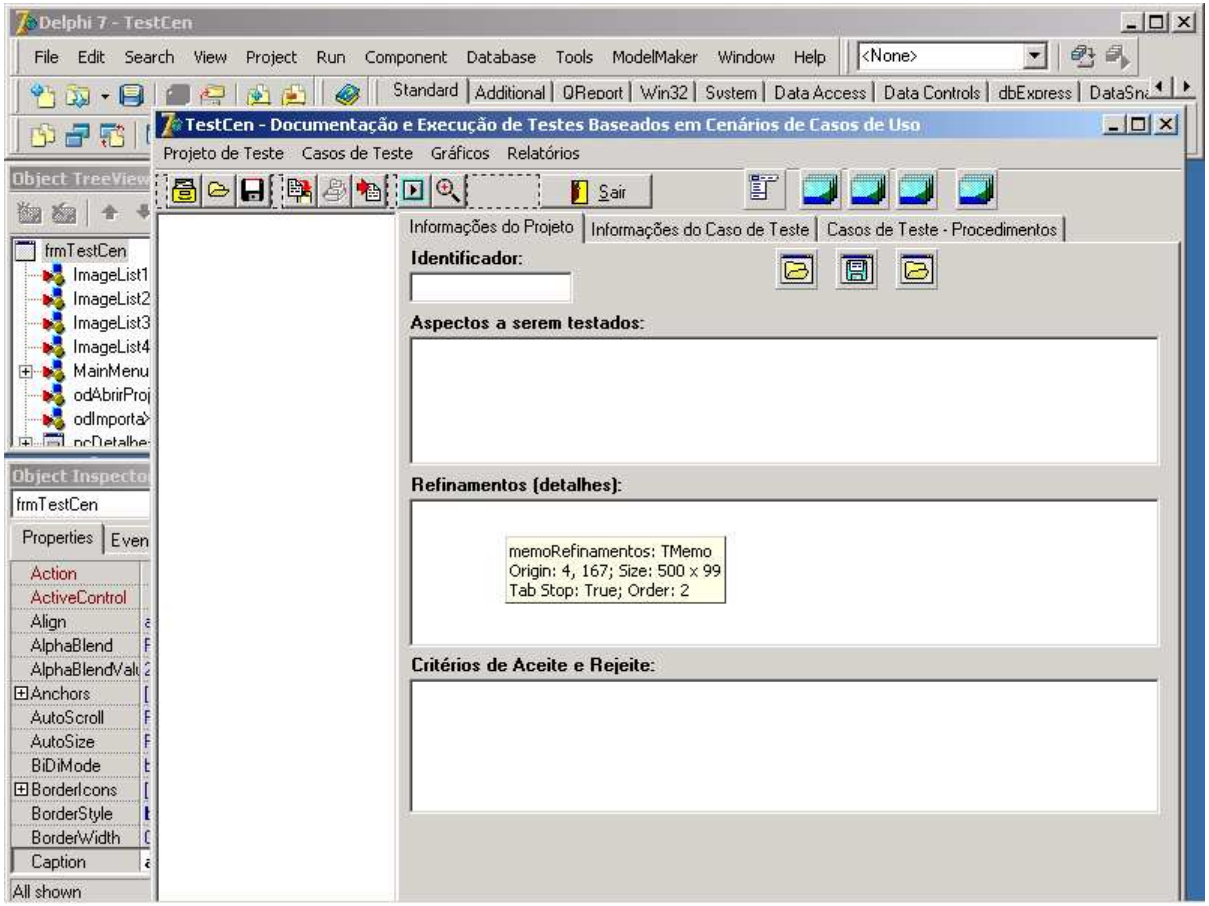
Implementação

- Alterações no ArgoUML:
  - Criação de duas guias para inserir informações para teste;



Implementação

- Implementação da ferramenta TesCen:



Técnicas e Ferramentas Utilizadas

- Modelagem UML com a ferramenta CASE ArgoUML;
- Para a customização da ferramenta ArgoUML foi utilizada a IDE NetBeans juntamente com o pacote de desenvolvimento Java™ 2 SDK Standard Edition versão 1.4.2;

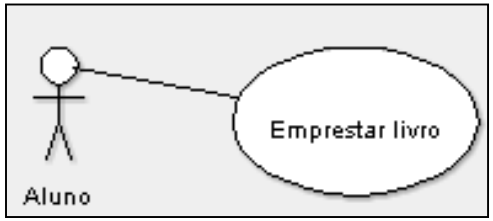
Técnicas e Ferramentas Utilizadas

- A ferramenta TesCen foi escrita em Object Pascal utilizando a IDE Delphi 7;

- Utilização de técnicas de análise e projeto de software orientado a objetos com UML (BEZZERA, 2002) e técnicas de teste funcional de software, além de conceitos de arquitetura de software e XML.

Operacionalidade da Implementação

- Elaboração de um conjunto de testes para um sistema fictício de controle de uma biblioteca;



Caso de uso emprestar livro	
Identificador	CU001
Ator primário	Aluno
Ator secundário	Bibliotecário
Pré-condições	O aluno deve estar cadastrado no sistema de acadêmicos e ter um cadastro e senha na biblioteca.
Fluxo Principal	O aluno informa os livros a serem emprestados. A bibliotecária insere o código de cada livro e o sistema registra a data e hora do empréstimo, o código do livro e qual a data de devolução do mesmo.
Fluxo Alternativo 1	O aluno tenta emprestar um livro, porém existem livros emprestados cuja data de entrega já venceu. O sistema não deve permitir o empréstimo.
Fluxo Alternativo 2	O aluno tenta emprestar um livro, mas existem multas não pagas de atraso na entrega de livros. O sistema não deve permitir o empréstimo.
Fluxo Alternativo 3	O aluno tenta emprestar um livro, mas o número máximo de empréstimos já excedeu. O sistema não deve permitir o empréstimo.
Pós-condições	Para o fluxo principal, o sistema deverá ter registrado com sucesso o empréstimo do livro. Para os fluxos alternativos o sistema deverá emitir mensagens de erro e/ou alerta.

Operacionalidade da Implementação

- Caso de teste para o fluxo principal:
  - 1) Analisando a descrição textual do caso de uso foi encontrado o fluxo principal (e os fluxos alternativos);
  - 2) Em seguida foram identificadas as variáveis operacionais:

Código do aluno	Código do livro	Data e hora do empréstimo	Data da devolução
001	005	Data e hora do momento do empréstimo	Data do momento de empréstimo + 7 dias

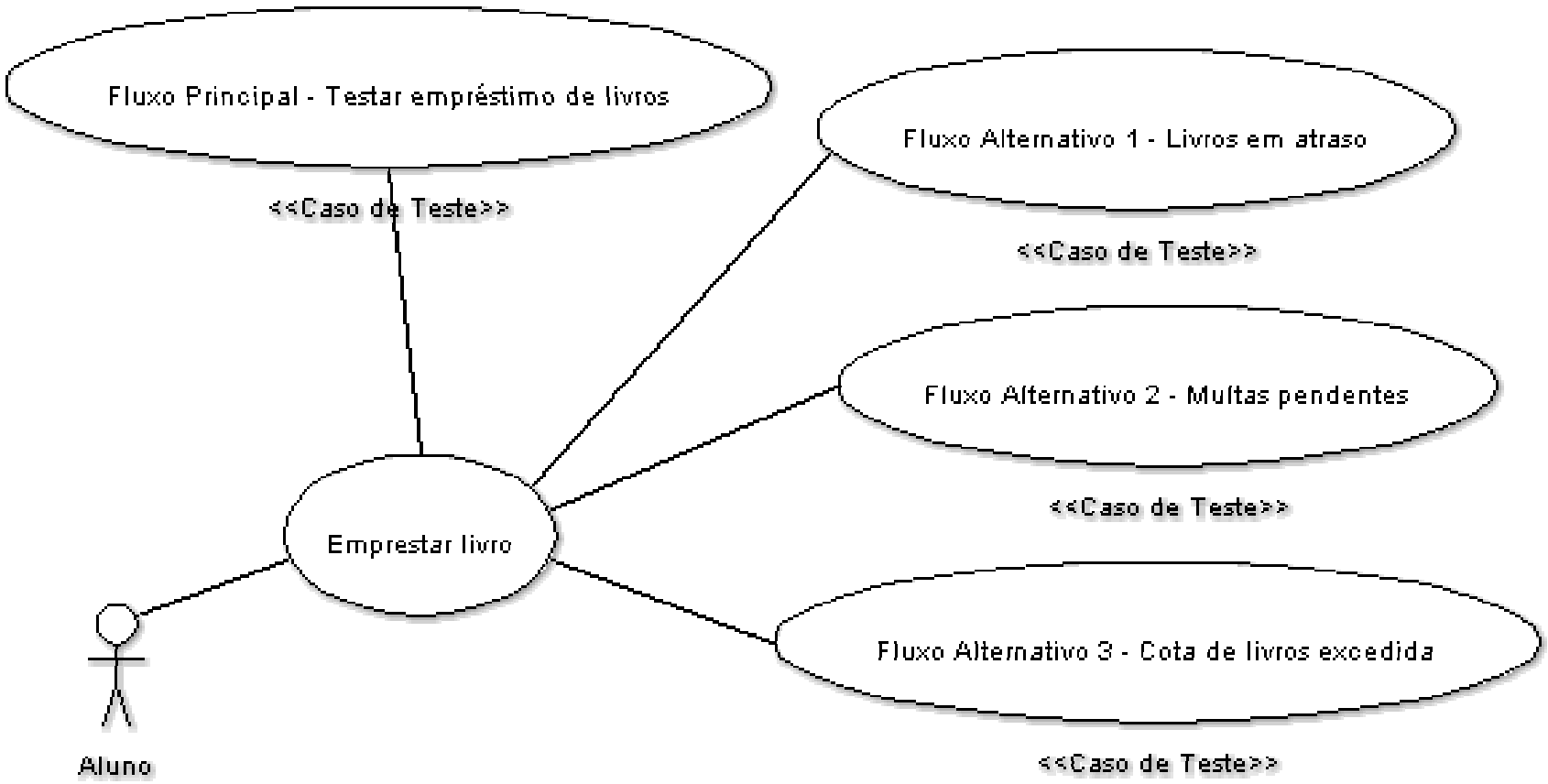
Operacionalidade da Implementação

- Passos:
  - 3) Desenvolvimento dos casos de teste, incluindo os procedimentos para execução. Estes foram desenvolvidos com base nas descrições dos casos de uso e nas variáveis operacionais identificadas.

Procedimentos	Verificações
Na tela de validação do aluno, digitar como código do aluno **001** e senha **abc123** (já previamente cadastrados).	O sistema deverá ir para a tela de empréstimo de livros.
Informar como código do livro a ser emprestado **005**.	O sistema deverá trazer as informações do livro: nome autor, descrição, editora e n° da edição.
Confirmar a empréstimo do livro para o aluno **001**.	O sistema deverá mostrar uma mensagem de confirmação do empréstimo, seguido de uma lista de livros emprestados pelo aluno, onde para o livro 005 a data e hora do empréstimo deve ser a data e hora do momento de empréstimo e a data de devolução deve ser a data do momento de empréstimo + 7 dias.

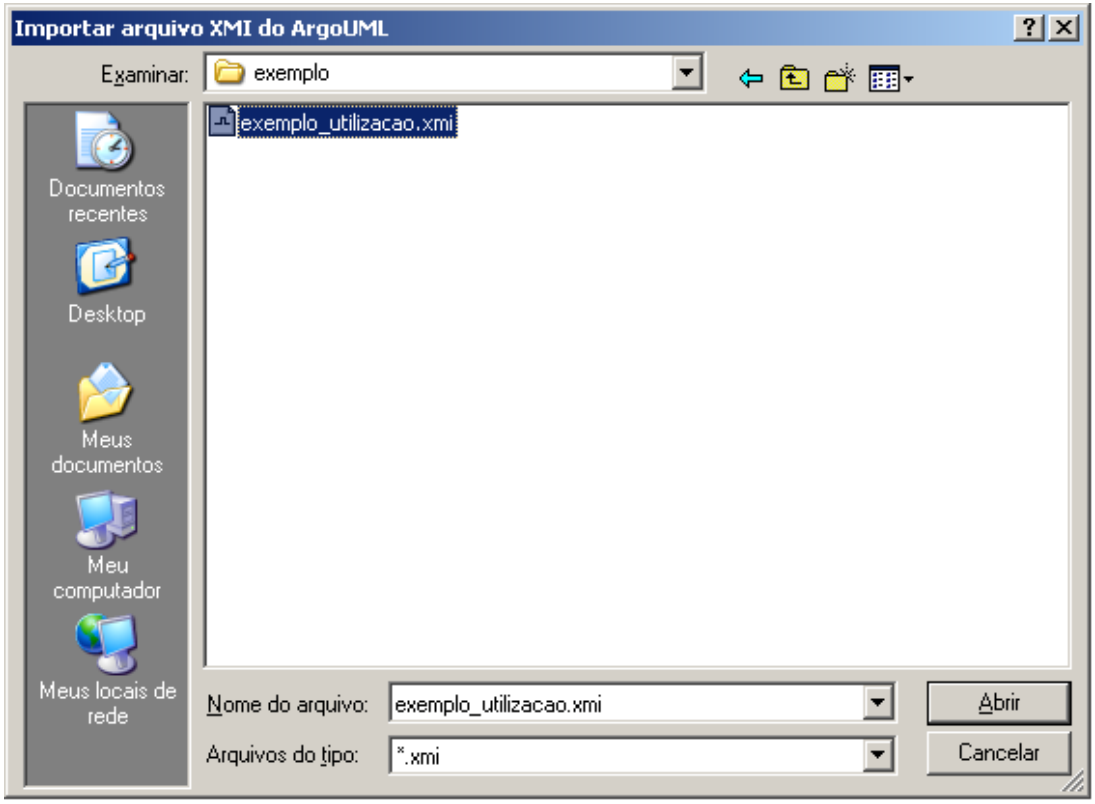
Operacionalidade da Implementação

- Especificação dos testes no ArgoUML:



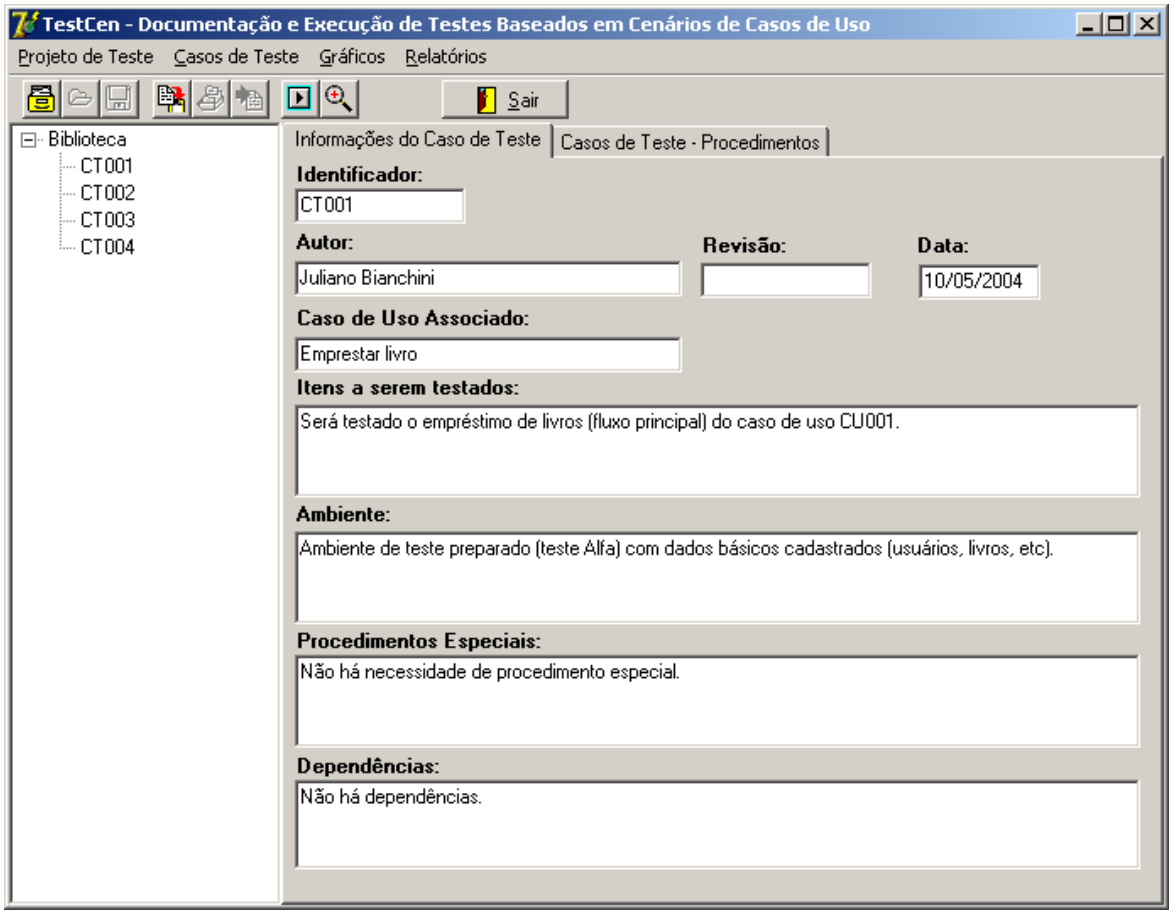
Operacionalidade da Implementação

- Importação dos dados do ArgoUML para a ferramenta TestCen:



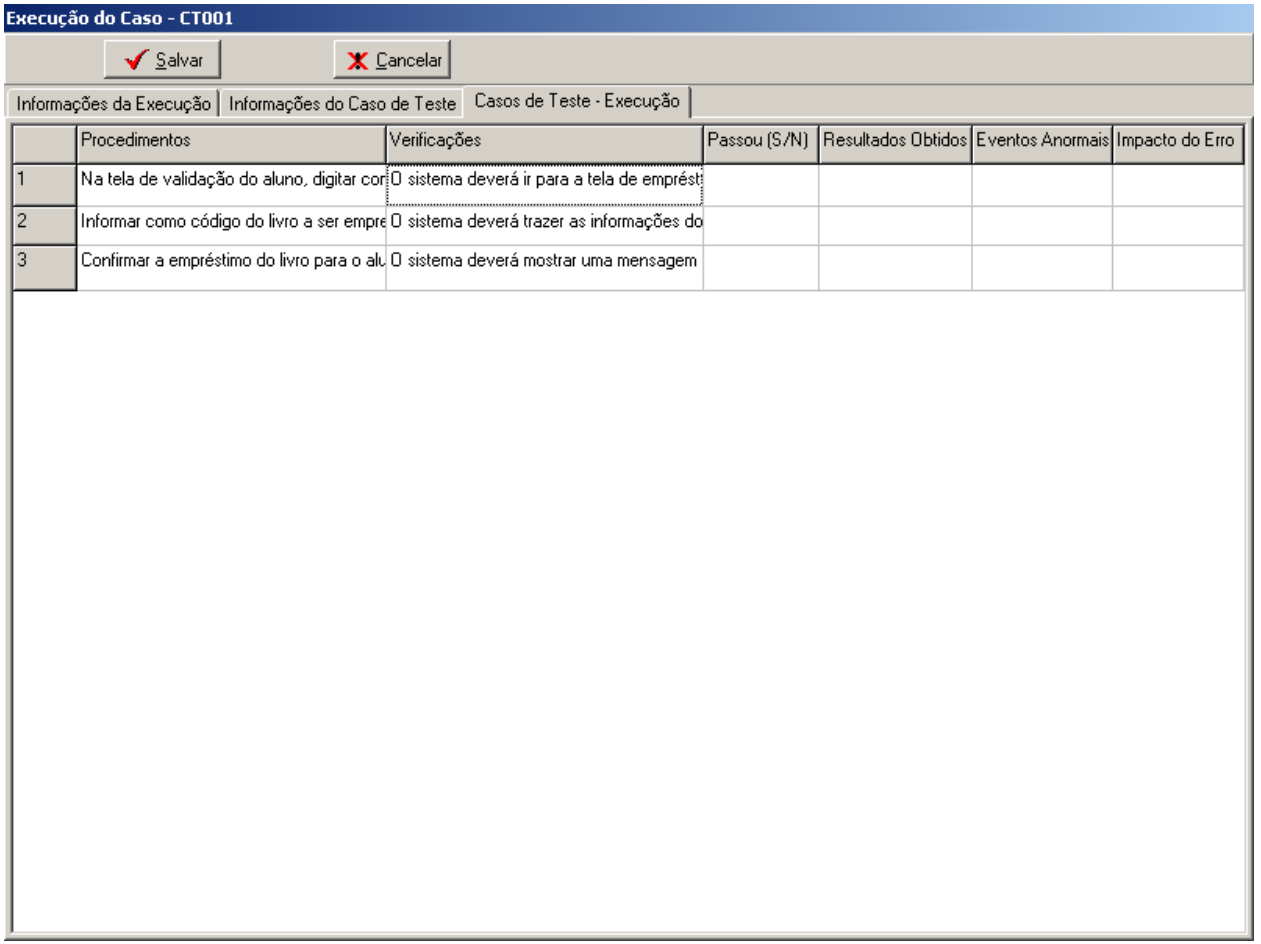
Operacionalidade da Implementação

- Testes especificados na ferramenta TestCen:



Operacionalidade da Implementação

- Execução de um caso de teste na ferramenta TestCen:



Resultados e Discussões

- A utilização das propostas de metodologias proporcionaram uma criação mais fácil e ágil dos casos de teste;

- A utilização das ferramentas facilitou e organizou todo o processo de teste, contribuindo também para a re-execução dos testes nas iterações/manutenções posteriores;

Resultados e Discussões

- A criação dos casos de teste é facilitada a medida que a especificação dos casos de uso é feita de forma detalhada;

- Dentre os documentos especificados pelo padrão IEEE 829 apenas a especificação do plano de teste e o relatório de transição de item de teste não foram implementados.

Conclusões e Extensões

- Verificação e validação dos requisitos de software feitos de forma simples e organizada;
- Desenvolvimento da ferramenta com base no padrão IEEE 829;
- Utilização das ferramentas facilitou e organizou o processo de planejamento e execução dos testes;

Conclusões e Extensões

- Este trabalho apenas supre parte do que a atividade de teste deve contemplar, outros tipos de teste devem ser aplicados;

- Os testes ficaram restritos ao orientando e seu orientador;

Conclusões e Extensões

- Como extensão, a implementação de um mecanismo de gravação e reprodução de entradas de dados em interfaces gráficas com o usuário;

- Outra extensão, seria incorporar um gerador de massa de dados, que possibilite a geração automática de dados de entrada, baseado em critérios estabelecidos, de forma a garantir massa de dados diferenciada para cada execução.

REFERÊNCIAS BIBLIOGRÁFICAS

BARTIÉ, Alexandre. **Garantia da qualidade de software**: adquirindo maturidade organizacional. Rio de Janeiro: Elsevier, 2002.

BEZERRA, Eduardo. **Princípios de análise e projeto de sistemas com UML**. Rio de Janeiro: Campus, 2002.

BINDER, Robert V. **Testing object-oriented systems:** models, patterns and tools. Addison-Wesley, 2000.

HEUMANN, Jim. **Generating test cases from use cases**. Disponível em: <http://therationaledge.com/content/jun_01/m_cases_jh.html>. Acesso em: 20 jan. 2004.

INSTITUTE OF ELECTRICAL AND ELECTRONIC ENGINEERS. **IEEE Std 829-1998**: IEEE standard for software test documentation. Nova York, 1998.

RYSER, Johannes; GLINZ, Martin. **A practical approach to validating and testing software systems using scenarios**. Disponível em: <http://www.ifi.unizh.ch/groups/req/ftp/papers/QWE99_ScenarioBasedTesting.pdf>. Acesso em: 01 set. 2003.

SEPIN. **Qualidade dos produtos de software**. Disponível em: <http://www.mct.gov.br/Temas/info/Dsi/Quali2001/QualiProutosSW2001.htm>. Acesso em: 15 abr. 2004.

SOMMERVILLE, Ian. **Engenharia de software**. 6. ed. São Paulo: Addison Wesley, 2003.

FIM

Obrigado!