COPPE/UFRJ

SELEÇÃO DE TÉCNICAS DE TESTE BASEADO EM MODELOS

Arilo Claudio Dias Neto

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Ciências em Engenharia de Sistemas e Computação.

Orientador: Guilherme Horta Travassos

Rio de Janeiro
Novembro de 2009

SELEÇÃO DE TÉCNICAS DE TESTE BASEADO EM MODELOS

Arilo Claudio Dias Neto

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

_____

Prof. Guilherme Horta Travassos, D.Sc.

_____

Prof[a]. Ana Regina Cavalcanti da Rocha, D.Sc.

_____

Prof[a]. Cláudia Maria Lima Werner, D.Sc.

_____

Prof[a]. Silvia Regina Vergílio, D.Sc.

_____

Prof. Márcio de Oliveira Barros, D.Sc.

RIO DE JANEIRO, RJ – BRASIL

NOVEMBRO DE 2009

Dias Neto, Arilo Claudio

Seleção de Técnicas de Teste Baseado em Modelos / Arilo Claudio Dias Neto – Rio de Janeiro: UFRJ/COPPE, 2009.

XV, 220 p.: il.; 29,7 cm.

Orientador: Guilherme Horta Travassos.

Tese (doutorado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2009.

Referências Bibliográficas: p. 172-177.

1. Teste de Software. Teste Baseado em Modelos. 3. Seleção de Tecnologias de Software. 4. Engenharia de Software Experimental. I. Travassos, Guilherme Horta II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Aos meus Pais, meus exemplos de vida e de onde tiro minha inspiração.
À minha Irmã, Danielly, à minha Sobrinha Lindsen e à Jullyana Campos.

Agradecimentos

Primeiramente a Deus, por estar comigo em todos os momentos e por ter me dado saúde e o dom da sabedoria. Nada seria possível sem Ele.

À minha Mãe, Roseneide, pelo amor, compreensão e apoio irrestrito em todos os momentos dessa longa jornada.

Ao meu Pai, José Ribamar, pelo carinho, força e por todo exemplo de vida que sempre me deu.

À minha Irmã, Danielly, e à minha sobrinha, Lindsen, pela força e inspiração. Quando deixei Manaus para vir ao Rio de Janeiro ela nem andada ou falava. Hoje já está uma mocinha com seus 6 anos.

À minha Namorada, Jullyana Campos, pelo amor e carinho. Sua preocupação e força dada a todo o momento foram muito importantes. Agora teremos toda a vida para comemorar esta data.

Aos meus Avôs, José Dirson, Arilo, Edite e Edith (em memória). Aos meus Tios, Abílio, Darlison, Dílson, Meire, Rose, Susana, Wálter, e Primos, Andrey, Caio, Gustavo, Maria Clara, Renan, Thamires, Thiago, Weyden, Weyne, pela torcida e apoio constante.

Ao meu Orientador, Guilherme Travassos, pela grande dedicação, conselhos e motivação durante todo este período, o que contribuiu não apenas para minha formação acadêmica, mas também minha formação como pessoa. Agradeço também pela orientação, incentivo e por me conduzir durante este trabalho e, por acreditar em mim e no meu trabalho.

Aos professores Ana Regina Rocha, Claudia Werner, Márcio Barros e Silvia Vergílio por participarem de minha banca de defesa de doutorado.

Especialmente aos amigos Rafael Barcelos e Rodrigo Spínola, pelo incentivo desde a minha chegada ao Rio até o dia de hoje. Tenho certeza que levarei essas amizades para sempre.

Aos Companheiros da COPPE, Beto, Fortuna, Jobson, Leonardo, Marco Antônio, Marcos, Paulo Sérgio, Taísa, Vitor e Wallace, pela amizade, sugestões e ajuda nos momentos que precisei. Aos alunos Lucas e Priscila pelo apoio na construção da infra-estrutura computacional proposta neste trabalho.

Aos alunos do curso de Engenharia de Software OO da graduação e da pós que aceitaram o convite e participaram dos estudos experimentais conduzidos neste trabalho.

À FAPEAM pelo apoio financeiro. À COPPE por prover a infra-estrutura.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

SELEÇÃO DE TÉCNICAS DE TESTE BASEADO EM MODELOS

Arilo Claudio Dias Neto

Novembro/2009

Orientador: Guilherme Horta Travassos

Programa: Engenharia de Sistemas e Computação

Este trabalho propõe uma abordagem desenvolvida a partir de uma metodologia científica baseada na condução de estudos secundários e primários que possui o objetivo de apoiar a seleção combinada de Técnicas de Teste Baseado em Modelos (TTBMs) para projetos de software considerando dois aspectos: (1) o grau de adequação entre TTBMs e as características de um projeto de software e (2) o impacto de mais de uma TTBMs em variáveis do processo de teste. Resultados de uma avaliação experimental indicam que esta abordagem contribui para melhorias na efetividade e eficiência do processo de seleção de TTBMs quando comparada a outra abordagem de seleção disponível na literatura técnica. Finalmente, é apresentada uma infra-estrutura computacional desenvolvida para apoiar a abordagem de seleção proposta, avaliada por engenheiros de software de uma organização de software internacional que indicaram que a infra-estrutura proposta contribui positivamente para o processo de seleção de TTBMs.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

SELECTION OF MODEL-BASED TESTING TECHNIQUES

Arilo Claudio Dias Neto

November/2009

Advisor: Guilherme Horta Travassos

Department: Computer Science and Systems Engineering

This work proposes an approach developed by following a scientific methodology based on the conduction of secondary and primary studies with the purpose of supporting the combined selection of Model Based Testing (MBT) techniques for a given software project considering two aspects: (1) the adequacy level between MBT techniques and the software project characteristics and (2) impact of more than one MBT technique in some testing process variables. Results of an experimental evaluation indicate this approach contributes to improve the MBT techniques selection process effectiveness and efficiency when compared to another selection approach available in the technical literature. Finally, it is presented a computerized infrastructure to support the proposed selection approach, evaluated by software engineers of an international software organization that indicated the infrastructure can contribute positively for the MBT selection process.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

CAPÍTULO 1 - INTRODUÇÃO

Neste capítulo serão apresentados o contexto do trabalho, o que motivou esta pesquisa e a questão de investigação. São também apresentados os seus objetivos, a metodologia de pesquisa adotada, o histórico deste trabalho e a organização deste texto.

1.1 Contexto e Motivação: Descrição do Problema

A seleção de tecnologias (ex: processos, produtos, técnicas, métodos, ferramentas que podem ser utilizadas para apoiar o desenvolvimento de software) a serem usadas em um dado projeto de software tem sido um tópico de pesquisa que começou a ser discutido com mais intensidade a partir de 1991, quando uma proposta de abordagem de apoio à seleção e reúso de diferentes tipos de tecnologias de software foi publicada por BASILI e ROMBACH (1991). Desde então, observa-se que a seleção de tecnologias de software é uma tarefa complexa que pode influenciar diretamente na efetividade do processo e qualidade do produto final (VEGAS e BASILI, 2005). De acordo com BERTOLINO (2004), a definição das técnicas mais adequadas para apoiar uma tarefa do desenvolvimento de software ainda é uma questão em aberto. Quanto mais informações forem providas para apoiar esta decisão, mais fácil se tornará o processo de seleção, minimizando assim a possibilidade de realizar escolhas inadequadas que possam resultar em um efeito negativo na qualidade do software.

O principal desafio relacionado a este problema é entender e decidir pela seleção de qual tecnologia de software melhor se adéqua para uma tarefa específica de um projeto de software. Assim, vários aspectos podem influenciar nesta decisão, tais como: conhecimento técnico e habilidade da equipe do projeto sobre tais tecnologias e sobre o domínio do problema, cronograma do projeto, esforço e custo associados ao uso de uma tecnologia, aspectos políticos dentro de uma organização de software, dentre outros.

No contexto da indústria de software, a decisão a respeito de qual tecnologia adotar pode um ponto de risco a ser gerenciado em um projeto de software. Existe atualmente uma ampla variedade de tecnologias disponíveis para apoiar o desenvolvimento de um software, e as principais razões pela qual engenheiros de

software não estão aptos a realizarem boas escolhas durante a seleção de tecnologias de software, segundo VEGAS e BASILI (2005), são:

- Informações sobre as diferentes tecnologias estão distribuídas em diferentes fontes (ex: livros, artigos científicos ou repositórios);
- Carência de diretrizes para apoiar a seleção de tais tecnologias para um projeto de software, e;
- Pouco conhecimento científico sobre tais tecnologias e sobre seu uso em projetos de software passados.

O problema da seleção de tecnologias pode ser definido como a composição de dois problemas:

- A completude do conjunto de tecnologias na qual a seleção será baseada;
- A identificação apropriada das características que representam simultaneamente as tecnologias e o ambiente do projeto de software.

Com respeito ao problema de completude, a composição do conjunto inicial de tecnologias terá uma influência decisiva na seleção final. Suponha que tenhamos dois conjuntos de tecnologias diferentes, sendo um considerado mais adequado de acordo com a avaliação de um membro da equipe a partir de um dado aspecto do projeto de software a ser desenvolvido. Com posse dessa informação, o membro da equipe do projeto utilizará apenas o conjunto mais adequado para então escolher a tecnologia que melhor se adéqua às características do projeto de software a ser desenvolvido, pois se não optar por este conjunto ele poderia escolher uma tecnologia não tão adequada (que estaria no conjunto menos adequado).

Atualmente, a razão pela qual um conjunto de tecnologias pode variar de um projeto para outro é devido principalmente ao conhecimento da pessoa responsável pela seleção, pois cada pessoa possui um determinado conhecimento sobre as diferentes tecnologias. Existem várias razões que justificam o fato de que as necessidades de desenvolvedores variam entre projetos, mas isso está sempre associado às características/requisitos do projeto, tais como o tipo de software a ser desenvolvido, ou às características da equipe de desenvolvimento (VEGAS e BASILI, 2005).

Para algumas tarefas do processo de desenvolvimento, a escolha de uma tecnologia de software usualmente é única, ou seja, somente uma tecnologia de software é suficiente, visto que a combinação de tecnologias com o mesmo propósito não introduziria melhores resultados ao processo de desenvolvimento de software. Por exemplo, a seleção de uma ferramenta de modelagem em um projeto de software.

Usualmente, apenas uma ferramenta de modelagem é necessária, desde que esta atenda a todas as necessidades de modelagem requeridas pelo projeto, pois a utilização de mais que uma ferramenta não traz benefícios ao processo, podendo inclusive aumentar o esforço e custo. A escolha de uma única tecnologia de software para apoiar em uma determinada atividade do processo de desenvolvimento torna a tarefa de selecionar tecnologias de software mais simples.

Por outro lado, para outras tarefas, a combinação de duas ou mais tecnologias de software com propósitos similares pode introduzir uma melhoria na efetividade do processo e, consequentemente, na qualidade do produto final. Um exemplo de atividade que poderia ser beneficiada pela combinação de diferentes tecnologias seria teste de software. No contexto de teste de software, a aplicação de mais de uma técnica simultaneamente para avaliação de um software pode resultar em uma maior cobertura dos testes e, consequentemente, aumento da qualidade do produto final. No entanto, este aumento de cobertura dos testes pode resultar em um aumento do esforço e custo para os testes caso as técnicas de teste selecionadas possuam pré-requisitos conflitantes para seu uso em um mesmo projeto. Portanto, a seleção de técnicas de teste precisa ser analisada cuidadosamente, pois elas podem inviabilizar as atividades de teste em um projeto de software (MENZIES *et al*., 2002).

As atividades de teste, por sua natureza, possuem características específicas quando comparadas com outras atividades do processo de desenvolvimento. Por ser uma atividade de garantia da qualidade e o último recurso para se avaliar um software antes de sua instalação em um ambiente de produção, seu sucesso depende exclusivamente da cobertura dos elementos do software atingida durante a sua realização, de forma a revelar a maior quantidade possível de falhas em um software antes de sua instalação em ambiente de produção ou estimar a sua confiabilidade a partir da não revelação de falhas (JURISTO *et al*., 2004).

Devido a esta necessidade de se garantir a qualidade dos testes, e consequentemente a qualidade do software, a aplicação de diferentes técnicas de teste que possibilitem avaliar diferentes características de qualidade ou níveis de abstração de um software é uma estratégia essencial para se atingir este objetivo. A aplicação de técnicas de teste que se complementam contribui para o aumento da cobertura dos testes, e com isso aumenta a confiabilidade do software. O Critério de Cobertura dos Testes permite a identificação de partes do programa que devem ser executadas para garantir a qualidade do software e indicar quando o mesmo foi suficientemente testado (RAPPS e WEYUKER, 1982), ou seja, possibilita determinar o percentual de elementos/partes de um software que foram executados pelo conjunto de casos de teste (ROCHA et al., 2001).

Pode ser observado ainda que cada subcategoria de técnicas de teste possui características específicas que podem influenciar no processo de seleção de técnicas de teste em um projeto de software.

Em 2005 iniciou-se uma parceria institucional entre o *SIEMENS Corporate Research* (SCR), sediado em Nova Jérsei nos Estados Unidos, e o Grupo de Engenharia de Software Experimental da COPPE/UFRJ. Na ocasião, optou-se por iniciar pesquisas cooperadas entre as duas instituições na área de Teste de Software, com um foco maior na subárea Teste de Software Baseado em Modelos.

Contextualizando o problema de seleção de tecnologias para a área de Teste Baseado em Modelos (TBM), podem ser observados desafios adicionais para a seleção de Técnicas de Teste Baseado em Modelos (TTBMs) para um projeto de software dadas as características específicas desta subcategoria de técnicas de teste que requerem diferentes preocupações quando comparadas com outras subcategorias de técnicas de teste. Entre os desafios para a seleção de tais técnicas, podem ser citados a identificação de quais características são de fato relevante para a seleção de tais técnicas em comparação a outras subcategorias de técnicas de teste, quais características são específicas para o domínio de TBM e quais são as principais atividades que impactam na aplicação destas técnicas em um projeto de software.

Teste Baseado em Modelos consiste em uma estratégia de teste na qual casos de teste são derivados total ou parcialmente a partir de modelos descrevendo algum aspecto (ex: funcionalidade, segurança, desempenho, etc.) de um software (UTTING e LEGEARD, 2007). Entre as características específicas da área de TBM, podem ser citadas a dependência de um modelo formal descrevendo o comportamento/estrutura do software que pode ser aplicado apenas para uma plataforma de software específica ou que requer uma equipe de teste com conhecimento sobre uma linguagem de modelagem.

Enquanto pesquisas científicas recentes produzem resultados interessantes no que diz respeito ao desenvolvimento de novas técnicas e infra-estruturas para apoiar TBM, por outro lado existe ainda uma carência de conhecimento científico sobre tais TTBMs já publicadas na literatura técnica, o que dificulta a transferência dessas tecnologias do ambiente acadêmico para a indústria. De acordo com DIAS-NETO *et al*. (2008), alguns fatores contribuem para este cenário:

- Alto número de TTBMs disponíveis na literatura técnica.
- Inexistência de um corpo de conhecimento ou repositório com informações sobre tais TTBMs.
- Carência de conhecimento científico (evidência) a respeito do uso de tais TTBMs em diferentes projetos de software.

Esses aspectos dificultam a seleção de apenas um subconjunto de TTBMs para um dado projeto de software, pois não existem atualmente informações suficientes providas que nos permita ter uma maior precisão sobre esta decisão. Atualmente, esta tarefa é realizada baseando-se em conveniência, ou seja, são aplicadas normalmente aquelas TTBMs já conhecidas por uma equipe de teste, independentemente de sua adequação para o projeto de software (VEGAS e BASILI, 2005).

1.2 Trabalhos Relacionados

Na literatura técnica, existem algumas abordagens que apóiam a seleção de tecnologias de software. Tais abordagens são direcionadas, principalmente, à seleção de tecnologias de software em geral e técnicas de apoio à identificação de requisitos, conforme descrito a seguir:

- **Seleção de Tecnologias de software em geral:**
  - (BASILI e ROMBACH, 1991): abordagem de seleção pioneira que se baseia na utilização de esquemas de caracterização de tecnologias de software baseados em modelos para se registrar o uso de tais tecnologias em projetos, viabilizando seu reúso em futuros projetos a partir de consulta a tais esquemas de caracterização das tecnologias previamente adotadas.

  - (BIRK, 1997): esquema de caracterização genérico para tecnologia de software baseado na modelagem do domínio de uma aplicação utilizando o paradigma GQM – *Goal-Question-Metric* (BASILI *et al*., 2004). Por utilizar a caracterizacao de técnicas de identificacao. Por não conter atributos de caracterização específicos para teste de software, sua aplicação para este contexto se torna difícil.

- **Seleção de Técnicas de Identificação de Requisitos:**
  - (MAIDEN e RUGG, 1996): abordagem para seleção de técnicas de identificação de requisitos baseada em questões a partir de um conjunto de características que definem os pontos fortes e fracos de cada técnica, provendo diretrizes a engenheiros de software para decidir pela escolha de técnicas de identificação de requisitos para um projeto. Por tratar apenas de características e tarefas relevantes para a atividade de identificação de requisitos, não pode ser aplicada no contexto proposto neste trabalho.

- (ARANDA *et al*., 2006): abordagem de seleção de tecnologias de software contextualizada para o cenário de desenvolvimento globalizado de software. Abordagem aplicada em atividades fortemente baseadas em comunicação, como identificação de requisitos, e propõe o uso de estratégias oriundas do campo de psicologia cognitiva para definir uma abordagem de seleção de tecnologias de software para um grupo de usuários e desenvolvedores atuando em um processo distribuído. Utiliza fatores não técnicos para apoiar tal seleção, o que dificulta sua aplicação para o contexto de teste de software.

Entre as abordagens citadas acima, nenhuma provê apoio à seleção combinada de mais de uma tecnologia de software para um mesmo projeto, ou seja, elas provêm apenas apoio à seleção individual de tais tecnologias. Além disso, por suas características específicas, elas não são aplicáveis para apoiar a seleção de técnicas de Teste de Software, que é o foco principal deste trabalho.

Durante a realização desta pesquisa, foram identificados 2 (dois) trabalhos principais relacionados à seleção de técnicas de teste para projetos de software utilizando dados e características técnicas para apoiar tal tomada de decisão.

- **Seleção de Técnicas de Teste:**
  - (VEGAS e BASILI, 2005): abordagem de apoio à seleção de técnicas de teste, chamada *Esquema de Caracterização*, na qual os atributos das técnicas de teste são armazenados em um repositório, e então para cada projeto de software é gerado um catálogo de técnicas de teste para apoiar na tomada de decisão. O trabalho de VEGAS e BASILI (2005) foi baseado em estratégias de seleção aplicadas em outras áreas como, por exemplo, reúso de componentes, e foi aplicado em testes de software. Os aspectos positivos deste trabalho são: a disponibilização de conhecimento sobre tais técnicas e a facilidade provida para a escolha de técnicas de teste de propósito geral para projetos de software. No entanto, a generalidade de seus atributos de caracterização dificulta a sua adequação para categorias específicas de técnicas de teste, tal como teste baseado em modelos, o que pode significar um risco para o projeto de software que esteja usando este tipo de abordagem para seleção de TTBMs. Outro aspecto restritivo desta abordagem é que esta não provê apoio à seleção combinada de mais de uma técnica de teste.

  - (WOJCICKI e STROOPER, 2007): abordagem de apoio à seleção de técnicas de Verificação e Validação (V&V) que avalia a combinação de técnicas de V&V com o objetivo de maximizar a completude das técnicas, representada pelo tipo de

defeitos/falhas revelados(as) pelas técnicas selecionadas, e minimizar o esforço, representado por métricas como o esforço da detecção de defeitos/falhas (em minutos) e efetividade de detecção de defeitos/falhas (quantidade de defeitos/falhas por tempo). Esta abordagem apresenta um processo para seleção de técnicas de V&V composto por três atividades (Pré-seleção, Maximizar Completude, Minimizar Esforço) utilizando matriz para relacionar os tipos de defeitos/falhas revelados(as) pelas técnicas e seu esforço. Tal matriz é preenchida passo a passo, utilizando métricas coletadas em uma organização e dados subjetivos. A principal vantagem desta abordagem é sua simplicidade para analisar uma possível melhor combinação de técnicas de V&V para um projeto. Além disso, como ela utiliza métricas de custo-eficiência, cada organização pode aplicar suas próprias métricas em diferentes níveis de detalhes durante o uso da abordagem.

A Tabela 1.1 apresenta um resumo das principais características das abordagens de apoio à seleção de tecnologias de software publicadas na literatura técnica e que foram citadas nesta seção. As características consideradas neste resumo foram:

- **Escopo:** a abordagem é aplicada à seleção de quais tecnologias de software?
- **Baseada em medição:** a abordagem provê informações de apoio à seleção baseando-se no uso de métricas coletadas a partir dos projetos/tecnologia de software?
- **Baseada em adequação:** a abordagem provê informações de apoio à seleção baseando-se na adequação entre a tecnologia de software e o projeto onde a tecnologia será aplicada?
- **Baseada no impacto no processo:** a abordagem provê informações de apoio à seleção baseando-se no possível impacto de tal tecnologia de software no processo de desenvolvimento que está sendo seguido?
- **Sugestão de tecnologias:** a abordagem sugere/indica formalmente quais tecnologias de software que mais se adéquam ao projeto de software a partir de algum critério pré-estabelecido?
- **Combinação de Tecnologias:** a abordagem analisa a combinação de duas ou mais tecnologias de software e sua influência na atividade a ser realizada, ou seja, provê apoio para a seleção combinada de tecnologias de software?
- **Apoio Ferramental:** existe algum apoio ferramental para se usar a abordagem?

Tabela 1.1. Caracterização de Abordagens de Apoio à Seleção de Tecnologias de Software

Abordagem de seleção	Basili & Rombach	Birk	Maiden & Rugg	Aranda *et al*.	Vegas & Basili	Wojcicki e Strooper
Escopo	Tecnologias de Software em geral	Tecnologias de Software em geral	Técnicas de Identificação de Requisito	Técnicas de Identificação de Requisito	Técnicas de Teste	Técnicas de Verificação e Validação
Baseada em Medição	NÃO	SIM	NÃO	NÃO	SIM	SIM
Baseada em Adequação	SIM	NÃO	NÃO	NÃO	SIM	NÃO
Baseada em Impacto	NÃO	NÃO	NÃO	NÃO	NÃO	SIM
Sugestão de Tecnologias	Provê um catálogo	Provê um catálogo	Analisa uma tecnologia indicada	Analisa uma tecnologia indicada	Provê um catálogo	Sugere tecnologias mais adequadas
Combinação de Tecnologias	Seleção Individual	Seleção Individual	Seleção Individual	Seleção Individual	Seleção Individual	SIM
Apoio Ferramental	Não encontrado	Não encontrado	Não encontrado	Não encontrado	SIM	Não encontrado

1.3 Questões de Pesquisa

Esta pesquisa propõe uma abordagem de apoio à seleção combinada de TTBMs para projetos de software. A questão de pesquisa para esse trabalho é baseada na questão formulada por BASILI e ROMBACH (1991) que diz:

> *O processo de seleção de tecnologias de software é melhorado usando mecanismos de caracterização de tecnologias?*

Então, o objetivo deste trabalho é avaliar uma possível instância estendida desta questão:

> *O uso de uma abordagem de seleção de técnicas de teste baseado em modelos que capture as particularidades destas técnicas melhora o processo de seleção e reduz os riscos associados à seleção de técnicas de teste de software baseada em modelos para serem usadas de forma combinada em um dado projeto de software em termos de eficiência, efetividade, completude, usabilidade e satisfação do usuário no processo de testes, comparada ao uso de outras estratégias mais genéricas ou à seleção baseada na percepção ou suposição de membros da equipe de teste de uma organização?*

A partir desta questão, algumas subquestões de pesquisa foram consideradas:

Q1) Seria possível melhorar o processo de seleção de TTBMs para projetos e organizações de software reduzindo o esforço, custo na utilização de uma ou mais técnica de forma combinada em um projeto e aumentando-se a cobertura dos requisitos de teste definidos em um projeto de software?

Q2) O uso de uma abordagem de apoio à seleção de TTBMs mais adequadas dadas as características de um projeto de software e que indique o impacto da combinação de tais técnicas em variáveis do processo de teste auxiliaria na melhoria da efetividade, eficiência, completude, usabilidade e satisfação do usuário durante o processo de seleção de TTBMs para projetos de software?

1.4 Objetivos: Estrutura da Solução

A seleção de tecnologias de software, independentemente a qual atividade do processo de desenvolvimento esteja sendo aplicada, pode depender de diversos fatores, dentre os quais podem ser citados fatores técnicos, tecnológicos, sociais, políticos e econômicos. A abordagem proposta neste trabalho utiliza como base alguns fatores técnicos (como a tecnologia funciona na prática) e tecnológicos (em qual contexto tecnológico esta pode ser aplicada) com o objetivo de prover informações que possam apoiar na tomada de decisão a respeito da seleção de TTBMs em projetos de software. Ela se utilizará de características técnicas e tecnológicas do projeto de software, no qual se pretende aplicar Teste Baseado em Modelos, e das TTBMs disponíveis em um repositório, a ser definido ao longo deste trabalho. É importante ressaltar que o objetivo principal deste trabalho é prover informações que apóiem na tomada de decisão a respeito de qual(is) TTBM(s) adotar em um projeto de software, provendo informações técnicas e tecnológicas a respeito de sua possível adequação a um projeto de software, e não substituir o papel do Gerente de Teste, que é o responsável pela decisão final, e este sim deve levar em consideração os demais fatores acima listados no processo de seleção de TTBMs para um determinado projeto de software.

Sendo assim, esta pesquisa irá focar em duas frentes para resolver o problema de seleção de tecnologias de software, contextualizando o problema para a área de Teste Baseado em Modelos: (1) fornecer conhecimento baseado em resultados de estudos experimentais para os profissionais responsáveis pela seleção sobre as possíveis opções associadas às TTBMs existentes e (2) identificar quais

características ou informações dos projetos de software e TTBMs têm influência na adequabilidade e impacto de uma ou várias TTBMs em um projeto de software.

A proposta desta pesquisa para o problema da seleção de TTBMs é o desenvolvimento de uma abordagem, batizada de *Porantim*[1], que avalia a adequabilidade e o impacto de TTBM a partir das características de um projeto de software onde elas seriam aplicadas. *Porantim* é composta por um corpo de conhecimento sobre TTBMs e por um processo de seleção responsável por guiar e prover informações para a equipe de teste executar esta tarefa.

1.4.1 Características do Corpo de Conhecimento sobre TTBM

O corpo de conhecimento a ser provido deve possuir as seguintes características, definidas nesta pesquisa:

- Deve ser específico para a caracterização de TTBMs. Ele não deve precisar de adaptação e deve estar pronto para ser instanciado para cada TTBM.
- As informações de cada TTBM devem ser providas incrementalmente no corpo de conhecimento, e devem ser constantemente atualizadas.
- O seu conteúdo e estrutura devem permitir atualizações sempre que necessário. Além disso, algumas informações devem ser atualizadas periodicamente a partir dos dados do uso das TTBMs em um projeto.

A definição detalhada deste corpo de conhecimento, incluindo sua estrutura e conteúdo, será apresentada no Capítulo 3 deste trabalho.

1.4.2 Características do Processo para Seleção de TTBMs

O processo responsável por guiar a equipe de teste na tarefa de seleção de TTBM deve possuir as seguintes características, também definidas nesta pesquisa:

- Deve prover informações individuais sobre a adequabilidade de cada TTBM em relação aos requisitos de um projeto de software previamente caracterizado.
- Deve indicar quais seriam as TTBMs mais adequadas ao projeto em questão, justificando tais indicações.

---

[1] *Porantim* é uma peça de Madeira usada pelos Índios Amazônicos *Sateré-Mawé* (que habitam a região do Baixo Amazonas) e que possui vários atributos: consiste em sua Bíblia e funciona como uma bola de cristal prevendo acontecimentos e resolvendo conflitos internos. É também usado no ritual de passagem *Wat'amã*, cujo objetivo é selecionar os índios que serão os guerreiros da tribo.

- Deve permitir que mais de uma TTBM seja selecionada para um mesmo projeto.
- Deve prover informações a respeito do impacto da seleção de mais de uma TTBM para os testes no projeto onde estas serão aplicadas.

A definição detalhada deste processo será apresentada no Capítulo 4 deste trabalho.

1.4.3 Cenário Proposto para *Porantim*

Porantim será fundamentada na abordagem desenvolvida por VEGAS e BASILI (2005) que consiste em um esquema de caracterização de técnicas de teste em geral, sendo incrementada para atender às características específicas da área de TBM, além das funcionalidades adicionais citadas anteriormente, algo não provido pela abordagem original de VEGAS e BASILI (2005).

O cenário proposto nesta pesquisa para *Porantim* é o seguinte (ver Figura 1.1):

- O papel da equipe de teste de uma organização (que serão chamados de *Consumidores*) será o de analisar e selecionar TTBMs para projetos de software, para que elas sejam usadas ao longo do projeto. As informações e conhecimento usados para essa seleção serão providos por *Porantim* através de um processo que irá consultar os dados das TTBMs diretamente em um repositório (corpo de conhecimento). Também é papel dos consumidores manter o repositório de TTBMs constantemente atualizado ao longo do tempo de acordo com os resultados que estes observaram a partir do uso das TTBMs em diferentes projetos na sua organização.
- O papel de pesquisadores em engenharia de software (que serão chamados de *Produtores de TTBM*) será o de criar novas TTBMs e alimentar o repositório com os dados iniciais sobre elas, assim como analisar as características e condições de aplicabilidade das técnicas. É difícil imaginar que cada pesquisador irá preencher um repositório com os dados de uma nova TTBM por ele desenvolvida. Por este motivo, um dos objetivos desta pesquisa é prover uma estrutura para viabilizar a manutenção do corpo de conhecimento de TTBMs com as técnicas publicadas na literatura técnica. Isso ocorrerá a partir da disponibilização de um protocolo de revisão sistemática cujo objetivo é identificar e caracterizar TTBMs publicadas na literatura técnica, provendo as informações necessárias para que estas sejam incluídas no repositório (corpo de conhecimento) provido nesta pesquisa. Este protocolo será descrito com maiores detalhes no Capítulo 2 deste trabalho.

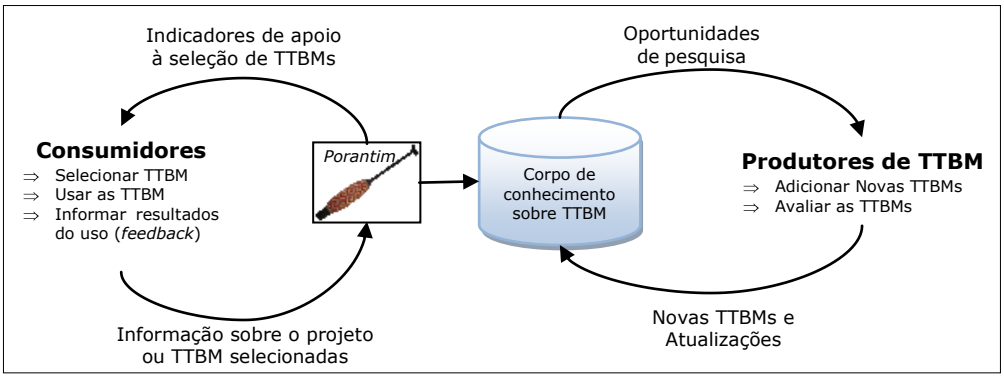


Figura 1.1. Cenário de uso proposto em *Porantim*

Os próximos capítulos irão descrever os elementos que compõem a solução proposta (corpo de conhecimento e processo de seleção) com maiores detalhes. A seguir, será descrita a metodologia de pesquisa seguida para se atingir a solução proposta. Serão detalhadas as etapas requeridas para o seu desenvolvimento, tipos de estudos a serem conduzidos e resultados esperados, a fim de prover resultados visando contribuir para o desenvolvimento da Engenharia de Software.

1.5 Metodologia de Pesquisa

A metodologia de pesquisa seguida para o desenvolvimento deste trabalho foi fundamentada na abordagem para desenvolvimento de novas tecnologias de software baseada em estudos primários e secundários publicada em SPÍNOLA *et al*., (2008). Ela se divide em duas fases: concepção e avaliação da abordagem proposta, conforme descrito na Figura 1.2.

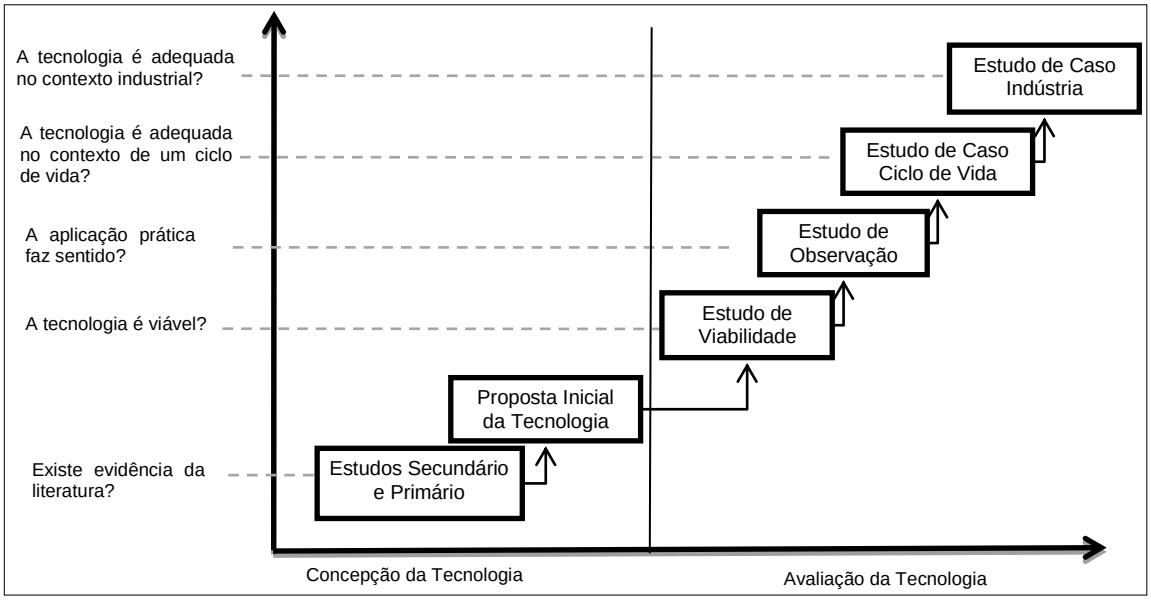


Figura 1.2. Metodologia de Pesquisa adotada (SPÍNOLA *et al*., 2008)

1.5.1 Fase de Concepção da Tecnologia

A fase de concepção da tecnologia envolve alguns passos e a execução de estudos secundários e/ou primários com o objetivo de se obter uma proposta inicial da tecnologia proposta.

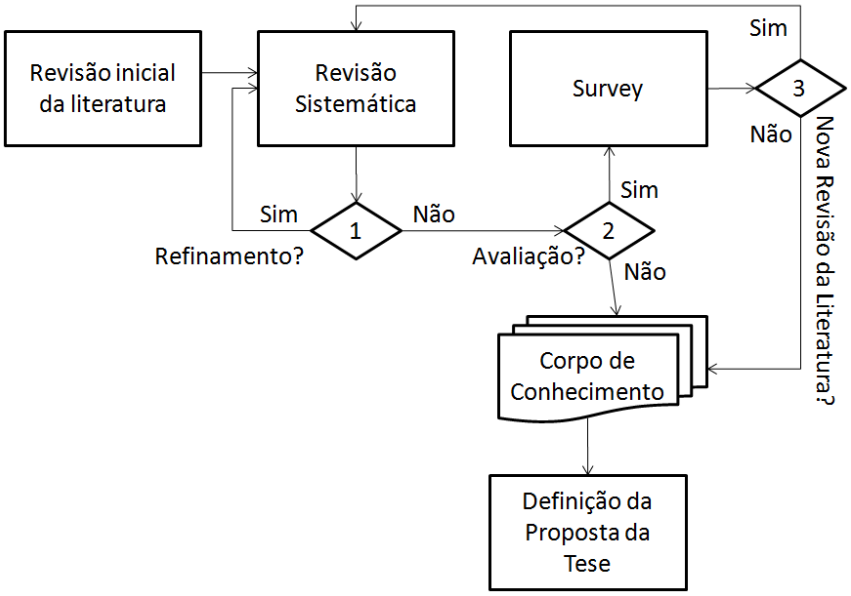


Figura 1.3. Metodologia de Pesquisa – Fase de Concepção (SPÍNOLA *et al*., 2008)

- **Revisão inicial da literatura:** executada de Março a Julho de 2006, nesta atividade realizou-se uma revisão *ad hoc* da literatura sobre a área de Teste Baseado em Modelos, seus conceitos, benefícios e processo. Esta pesquisa incluiu a busca de atributos relevantes para a caracterização de técnicas e infra-estrutura de apoio a TBM que seriam trabalhos relacionados ao trabalho proposto neste documento.
- **Revisão sistemática:** os fundamentos da revisão sistemática (estudo secundário) auxiliam na obtenção de resultados mais justos e precisos sobre o assunto que se pretende pesquisar (BIOLCHINI *et al*., 2005). Executada inicialmente no período de Julho a Dezembro de 2006 e re-executada no período de Julho a Agosto de 2009, nesta atividade realizou-se uma revisão controlada da literatura cujo objetivo foi identificar e caracterizar TTBMs publicadas na literatura técnica extraindo de cada técnica suas informações de acordo com os atributos de caracterização de TTBM identificados no passo anterior da metodologia. Os resultados deste estudo serão apresentados no Capítulo 2 deste trabalho.

- ***Survey*:** executada de Setembro de 2007 a Fevereiro de 2008, o objetivo desta atividade foi avaliar os atributos que descrevem TTBM extraídos a partir dos passos anteriores da metodologia com o propósito de caracterizá-los com respeito a sua importância para caracterizar uma TTBM e sua relevância no contexto da seleção de TTBMs para projetos de software. O resultado de sua execução foi a geração do corpo de conhecimento sobre TTBMs de forma mais estruturada e com resultados baseados na opinião de especialistas. Os resultados deste estudo serão apresentados no Capítulo 3 deste trabalho.
- **Definição da Proposta de Tese:** executada de Março de 2008 a Novembro de 2008, com o corpo de conhecimento estabelecido, passou-se a trabalhar na abordagem proposta para apoiar a seleção de TTBMs para projetos de software. Os resultados desta etapa serão apresentados no Capítulo 4 deste trabalho.

1.5.2 Fase de Avaliação da Tecnologia

Concluída a fase de concepção, passou-se à etapa de avaliação da abordagem proposta. Para tal fase, foram realizados dois estudos com diferentes propósitos:

- **Estudo de Viabilidade:** executado de Novembro a Dezembro de 2008, este estudo experimental teve como objetivo avaliar diversos aspectos (eficiência, efetividade, completude, usabilidade e satisfação do usuário) da abordagem proposta neste trabalho em relação a outra abordagem de apoio à seleção de técnicas de teste proposta por VEGAS e BASILI (2005). Os resultados deste estudo serão apresentados no Capítulo 5 deste trabalho.

- **Estudo de Observação:** executado de Maio a Junho de 2009, este estudo de avaliação teve como objetivo avaliar o comportamento da infra-estrutura computacional provida para apoiar a abordagem proposta neste trabalho com engenheiros de software que atuam no seu dia-a-dia com atividades de TBM.

A Figura 1.4 apresenta um resumo das etapas que compõem a metodologia de pesquisa adotada neste trabalho, assim como os tipos de estudos conduzidos e os resultados obtidos.

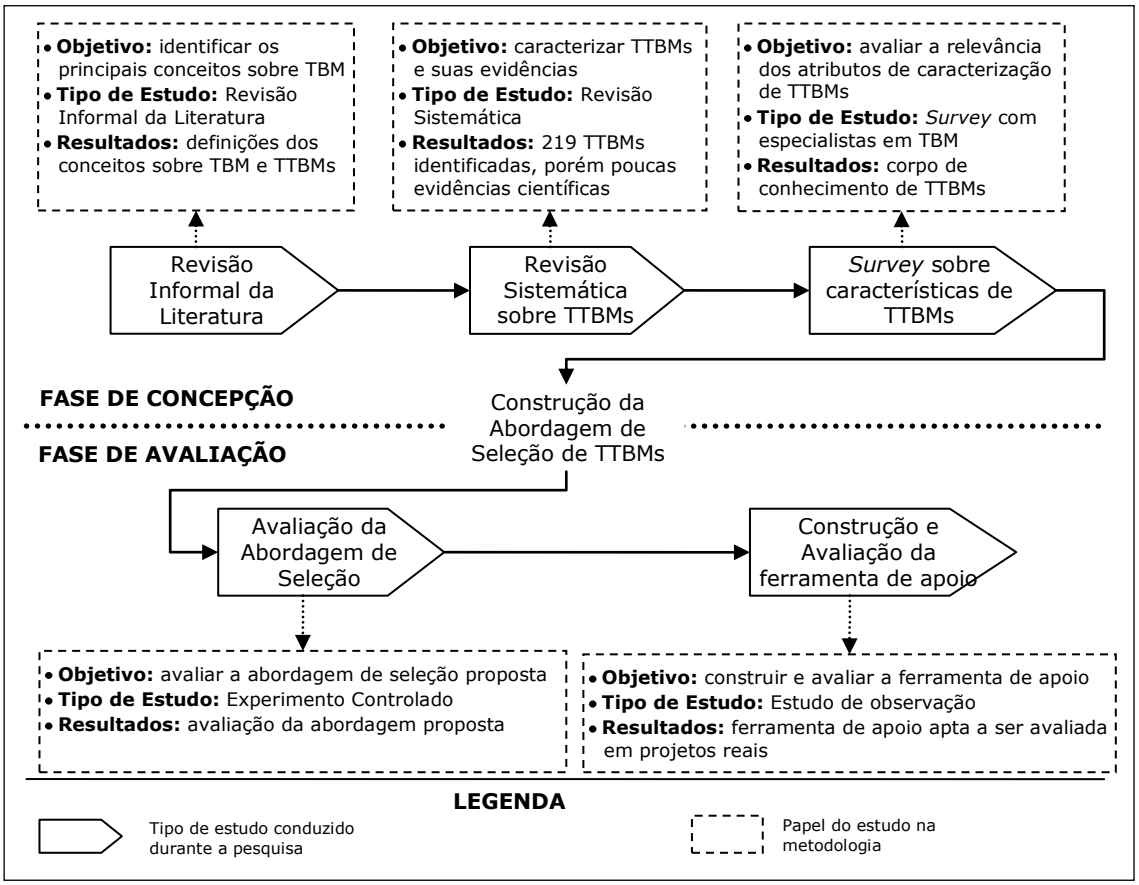


Figura 1.4. Resumo dos Estudos conduzidos ao longo desta pesquisa

1.6 Histórico da Pesquisa

Esta seção descreve o histórico desta pesquisa, indicando as atividades realizadas e as pesquisas publicadas ao longo deste período.

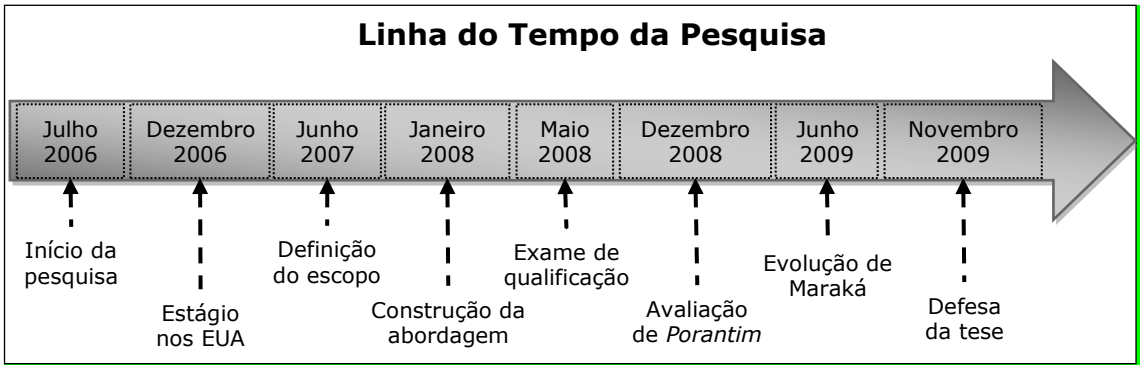


Figura 1.5. Linha do tempo da pesquisa

- **Início da Pesquisa – Maio/2006 a Julho/2006:** Início da pesquisa, quando foi escolhido pesquisar algum tema relacionado a Teste Baseado em Modelos a partir do início de uma parceria entre o grupo ESE da COPPE e

SCR (*SIEMENS Corporate Research*) e o convite para realizar um estágio supervisionado nos EUA.

- **Estágio nos EUA – Agosto/2006 a Dezembro/2006:** Realização do estágio supervisionado no SCR sediado em Princeton – Nova Jersey – EUA, cuja tarefa foi realizar uma pesquisa na literatura técnica (revisão sistemática) para identificar TTBMs similares à TTBM desenvolvida pelo grupo de teste de software do SCR.

- **Definição do Escopo – Janeiro/2007 a Julho/2007:** Observação de alguns problemas e oportunidades de pesquisa na área de TBM. Definição do escopo do trabalho: fornecer apoio à seleção de TTBMs em projetos de software.

- **Construção da Abordagem – Agosto/2007 a Janeiro/2008:** Construção do corpo de conhecimento de TTBMs e definição de uma versão inicial da abordagem de apoio à seleção de TTBMs proposta nesta pesquisa, batizada de *Porantim*.

- **Exame de Qualificação – Fevereiro/2008 a Maio/2008:** Preparação para o exame de qualificação do doutorado na COPPE/UFRJ. Participaram da banca de defesa os professores D.Sc. Cláudia Maria Lima Werner (COPPE/UFRJ), Dr. José Carlos Maldonado (ICM/USP) e D. Sc. Guilherme Horta Travassos (COPPE/UFRJ).

- **Avaliação de *Porantim* – Junho/2008 a Dezembro/2008:** Evolução da abordagem proposta e realização de estudos experimentais com o objetivo de avaliá-la em relação a outra abordagem de apoio à seleção de técnicas de teste com alunos de graduação e pós-graduação.

- **Evolução de Maraká – Janeiro/2009 a Junho/2009:** Extensão da infra-estrutura computacional Maraká para apoiar a seleção de TTBMs utilizando a abordagem *Porantim* proposta neste trabalho. Em seguida, tal infra-estrutura foi avaliada com engenheiros de software que atuam com TBM em projetos reais no SCR, Princeton-EUA.

- **Defesa da Tese – Julho/2009 a Novembro/2009:** Atualização do corpo de conhecimento de TTBMs a partir da re-execução da revisão sistemática realizada inicialmente no período de Julho/2006 a Dezembro/2006. 193 novos artigos foram identificados e analisados, e suas caracterizações foram adicionadas ao corpo de conhecimento de TTBMs disponibilizado nesta pesquisa. Por fim, foi realizada a escrita desta Tese de Doutorado (este documento) apresentada em Novembro/2009 na COPPE/UFRJ.

1.7 Organização do Trabalho

Este capítulo introdutório apresentou o contexto que motivou o desenvolvimento desta tese, as questões de pesquisa, a solução proposta, a metodologia de pesquisa adotada e um histórico desta pesquisa. Estes tópicos serão refinados ao longo dos próximos capítulos e apêndices. A organização do texto desta tese segue a estrutura abaixo:

- **Capítulo II – Teste Baseado em Modelos**: apresenta as principais definições sobre Teste Baseado em Modelos, seus benefícios ao processo de desenvolvimento de software e principais desafios. Em seguida, é apresentada uma revisão sistemática caracterizando TTBMs existentes na literatura técnica, e que serão utilizadas como conteúdo para o corpo de conhecimento de TTBMs proposto nesta pesquisa.
- **Capítulo III – Corpo de Conhecimento sobre Técnicas de Teste Baseado em Modelos**: apresenta o primeiro elemento que compõe a abordagem de apoio à seleção de TTBMs proposta nesta pesquisa: o corpo de conhecimento de TTBMs. São apresentados seu conteúdo e estrutura, resultados de estudos secundários e primários conduzidos ao longo deste trabalho.
- **Capítulo IV – *Porantim*: Uma Abordagem para Apoiar a Seleção Combinada de Técnicas de Teste Baseado em Modelos**: apresenta passo-a-passo o processo de seleção de TTBMs que compõe a abordagem *Porantim*, descrevendo as fórmulas e indicadores providos por *Porantim*.
- **Capítulo V – Avaliação Experimental da Abordagem *Porantim***: descreve um estudo experimental conduzido com o propósito de avaliar a abordagem proposta neste trabalho em relação a outra abordagem de apoio à seleção de técnicas de teste, cujo resultado indica que *Porantim* provê melhorias para a efetividade e eficiência do processo de seleção de TTBMs.

- **Capítulo VI – Infra-estrutura Computacional de Apoio à Seleção de TTBMs**: é apresentada a infra-estrutura computacional desenvolvida para apoiar na aplicação da abordagem *Porantim*, seus requisitos, arquitetura e funcionalidades, além dos resultados de um estudo para avaliação da infra-estrutura.
- **Capítulo VIII – Considerações Finais**: contém as considerações finais sobre esta pesquisa, seus resultados obtidos, as limitações do trabalho e futuros direcionamentos para a continuidade desta pesquisa.
- **Apêndice A – Fontes e Artigos Utilizados na Revisão Sistemática sobre Técnicas de Teste Baseado em Modelos**: apresenta a lista de fontes de artigos (bibliotecas digitais, sites, conferências) e a lista de artigos identificados a partir da revisão sistemática sobre TTBMs descrita no Capítulo 2 desta pesquisa.
- **Apêndice B – Plano do Estudo Experimental para Avaliação de *Porantim***: descreve o plano do estudo experimental realizado ao longo deste trabalho que avaliou a abordagem *Porantim*, descrito no Capítulo 5.
- **Apêndice C – Plano do Estudo de Avaliação de *Porantim* + Apoio Ferramental**: descreve o plano do estudo de avaliação realizado ao longo deste trabalho, e descrito no Capítulo 6, que avaliou a infra-estrutura computacional desenvolvida para apoiar a aplicação da abordagem *Porantim*.

CAPÍTULO 2 - TESTE BASEADO EM MODELOS

Neste capítulo estão listados os conceitos relacionados a teste de software baseado em modelos, que consiste na principal área de pesquisa abordada neste trabalho, e apresenta os resultados de uma revisão sistemática realizada para identificar e caracterizar Técnicas de Teste Baseado em Modelos publicadas na literatura técnica.

2.1 Introdução

Teste de Software consiste em uma investigação experimental conduzida para prover informações aos usuários e envolvidos no processo sobre a qualidade do software sob teste no contexto no qual este será operado. Isso inclui, mas não consiste apenas de um processo de executar um programa com a intenção de revelar falhas (KANER, 2006).

A definicao de falhas, citada acima, segue a terminologia padrão para Engenharia de Software do IEEE – *Institute of Electrical and Electronics Engineers* – (IEEE 610, 1990), onde são classficiados os termos *defeito*, *engano*, *erro* e *falha*.

- **Defeito (*fault*)** é um ato inconsistente cometido por um indivíduo ao tentar entender uma determinada informação, resolver um problema ou utilizar um método ou uma ferramenta. Como, por exemplo, uma instrução ou comando incorreto.
- **Engano (*mistake*)** é uma ação humana que produz um resultado incorreto, como uma ação incorreta tomada pelo programador.
- **Erro (*error*)** é uma manifestação concreta de um defeito num artefato de software. Diferença entre o valor obtido e o valor esperado, ou seja, qualquer estado intermediário incorreto ou resultado inesperado na execução de um programa constitui um erro.
- **Falha (*failure*)** é o comportamento operacional do software diferente do esperado pelo usuário. Uma falha pode ter sido causada por diversos erros e alguns erros podem nunca causar uma falha.

Sendo assim, teste está relecionada à atividade de revelar falhas em um sistema a partir de sua execução.

De acordo com JURISTO *et al*. (2004), Teste de Software pode ser considerado uma das atividades mais custosas no processo de desenvolvimento, e um dos principais fatores que influenciam no custo dos testes consiste na quantidade total de casos de teste identificados para um software, pois devem ser alocados recursos (físicos e humanos) para a geração e execução de cada caso de teste.

A partir dessas preocupações relacionadas ao planejamento e projeto dos testes, torna-se interessante pensar em como simplificar ou automatizar a execução e regressão (re-execução após alterações no software ou em sua especificação) dos testes, pois a atividade de geração dos casos de teste seria aquela que mais impacta a cobertura dos testes.

ROCHA *et al*. (2001) descrevem 3 estratégias de teste que podem ser aplicadas em projetos de software:

- Baseadas em especificação: avalia o produto gerando testes a partir de sua especificação, descrita em alguma linguagem formal ou textual.
- Baseadas em modelos: avalia o produto gerando testes a partir de modelos formais descrevendo comportamentos ou estrutura de um sistema. Tenta se antecipar à geração dos testes antes do sistema está desenvolvido.
- Baseadas em código: Tenta-se testar cada linha de código de um sistema já desenvolvido.

Além disso, basicamente 3 tipos de técnicas de teste podem ser aplicados:

- Técnica Funcional: aborda o software de um ponto de vista macroscópico e estabelece requisitos de teste com base na especificação.
- Técnica Estrutural: estabelece os requisites de teste com base emu ma determinada implementacao, verificando se atende aos detalhes do código.
- Técnica Baseada em Erros: estabelece os requisitos de teste explorando os erros típicos e comuns cometidos durante o desenvolvimento de software.

Teste Baseado em Modelos (TBM) surge como uma estratégia viável para controlar a qualidade do software, reduzindo os custos relacionados ao processo de testes, visto que os casos de teste podem ser gerados a partir de artefatos (modelos) produzidos ao longo do processo de desenvolvimento de software. A exploração de TBM possibilita ao Engenheiro de Software a realização de uma verificação adicional dos modelos produzidos ao longo do desenvolvimento de software antes que estes sejam passados a fases seguintes do processo, uma vez que estes modelos serão

utilizados para a geração dos testes e eles precisam estar corretos para garantir o sucesso desta atividade. Dessa forma, TBM contribui para a qualidade do software não apenas através da execução dos casos de testes após o desenvolvimento do software, mas também com a verificação dos modelos durante o processo de geração dos casos de testes.

2.2 Conceitos Básicos

Teste Baseado em Modelos consiste em uma estratégia de teste na qual casos de teste são derivados totalmente ou parcialmente de um modelo que descreve algum aspecto (ex: funcionalidade, segurança, desempenho, etc.) de um software (UTTING e LEGEARD, 2007). Para sua utilização, é necessário que o comportamento ou estrutura do software (ou parte deles) tenha sido formalizado através de modelos com regras bem definidas (tais como métodos formais, máquinas de estado finito, diagramas UML, dentre outros).

A utilização de modelos como forma de representação de um sistema para geração e execução de testes tem início praticamente num mesmo período que a própria área de teste de software. RAMAMOORTHY *et al*., em 1976, publicaram um artigo científico seminal sobre uma técnica de teste de software baseada em modelos que adotava o critério de fluxo de dados para programas em FORTAN. Em 1978, CHOW publicou um artigo científico que descreve uma técnica de teste baseada em modelos que adotava o critério de fluxo de controle para avaliar a corretude de um software a partir de sua representação através de uma máquina de estado finito. A evolução da estratégia de teste baseada em modelos tem ocorrido ano a ano. Em (DIAS-NETO *et al.,* 2008) é descrito um *survey* que caracterizou 71 TTBMs publicadas na literatura técnica a partir de 1990. Tais técnicas estão listadas no Apêndice A – seção AP A.3 neste trabalho, e que serão analisadas com maiores detalhes no Capítulo 2 - 2.3.

Apesar do fato de que TBM pode ser confundido com Geração de Casos de Teste, é preciso ressaltar a diferença entre essas definições para facilitar o entendimento. TBM usa modelos desenvolvidos durante qualquer fase do processo de desenvolvimento do software e ampliados pela equipe de teste para gerar automaticamente o conjunto de casos de teste. Em contrapartida, Geração de Casos de Teste é apenas uma das tarefas que compõem o processo de testes, e pode ser realizada ou não utilizando modelos de software formalizados.

A Figura 2.1 descreve as atividades específicas associadas a TBM, que são:

1. Construir o modelo descrevendo a estrutura/comportamento do software (uma das principais diferenças em relação às demais estratégias)
2. Geração de Casos de Teste
    a. Gerar entradas esperadas
    b. Gerar resultados ou comportamentos esperados
3. Executar os testes
4. Comparar os resultados obtidos com os resultados esperados
5. Decidir as futuras ações (se modificar o modelo, gerar mais testes, parar os testes ou estimar a confiabilidade – qualidade – do software)

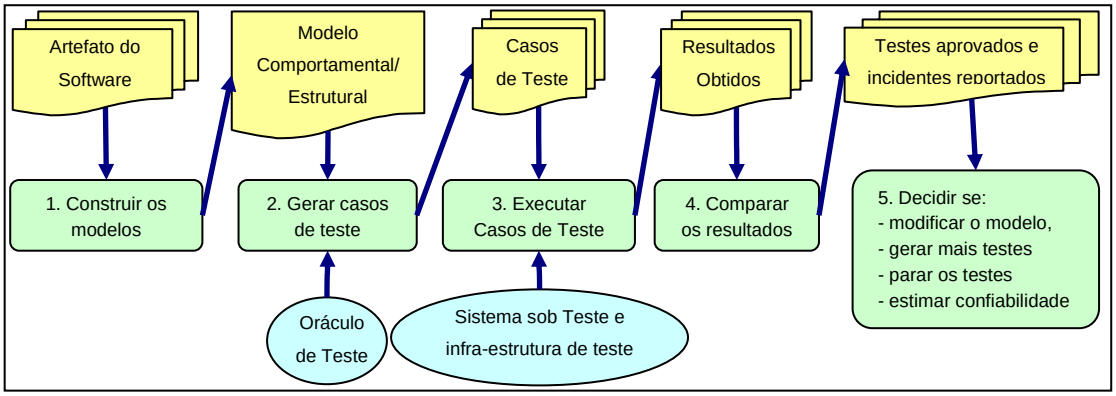


Figura 2.1. Atividades de TBM (adaptado de PRASANNA *et al*., 2005)

A estratégia de TBM normalmente inclui diferentes níveis de abstração, um modelo comportamental/estrutural do software, o relacionamento entre modelos e código, uma tecnologia para geração dos casos de teste, a importância dos critérios de seleção dos testes (algoritmos para geração dos casos de teste) e uma discussão do que pode ou não ser automatizado durante os testes (PRETSCHNER, 2005).

De acordo com UTTING e LEGEARD (2007), uma Técnica de TBM (TTBM) consiste em uma instância da estratégia de TBM apresentada na Figura 2.1 e pode ser aplicada a qualquer tipo de técnica de teste (funcional, estrutural ou baseada em erros). Diversas TTBM têm sido propostas na literatura técnica. Elas usam diferentes modelos para especificar o software, e esses modelos normalmente descrevem diferentes características de um produto. Isso torna a identificação, seleção e utilização de uma TTBM em um projeto de software uma tarefa complexa e de difícil decisão.

Quando se olha especificamente para a área de Teste Baseado em Modelos, percebe-se que enquanto pesquisas recentes têm produzido resultados interessantes a respeito do desenvolvimento de novas técnicas e infra-estruturas computacionais para apoiar TBM, existe ainda uma carência por conhecimento científico sobre tais

técnicas, o que dificulta na sua transferência para a indústria. Alguns fatores, identificados a partir de um estudo secundário, contribuem para este cenário:

- Alto número de TTBMs disponível na literatura técnica;
- Carência de um corpo de conhecimento ou repositório com informações sobre tais técnicas, e;
- Carência de conhecimento científico (evidências) obtido a partir do uso de TTBMs em diferentes projetos de software.

Esses aspectos tornam a seleção de TTBMs para um projeto de software uma tarefa difícil, devido à falta de informações que permitam aos engenheiros de software minimizar os riscos associados a esta decisão. Atualmente, esta tarefa tem sido realizada por conveniência da equipe de teste em um projeto de software, ou seja, são escolhidas as técnicas já conhecidas pela equipe, independentemente de sua adequação ao projeto de software (VEGAS e BASILI, 2005).

2.3 Revisão Sistemática sobre Técnicas de Teste Baseado em Modelos

Com o propósito de identificar e caracterizar TTBMs publicadas na literatura técnica, foi planejada e executada uma revisão sistemática. Sendo assim, um protocolo de busca e análise foi adotado. Ao longo desta pesquisa, este protocolo foi executado em dois momentos diferentes:

- **Julho a Dezembro de 2006:** nesta primeira execução, foram identificadas TTBMs publicadas entre o período de 1990[2] até 2006 (ano em que o protocolo foi executado). Ao total foram identificados inicialmente 406 artigos, reduzidos para 202 artigos após a aplicação de critérios de inclusão/exclusão. A descrição completa e detalhada, o protocolo de busca adotado, os critérios de inclusão/exclusão de artigos e a análise detalhada dos resultados obtidos nesta primeira rodada de execução do protocolo de revisão sistemática foram publicadas em (DIAS-NETO *et al*., 2007a), (DIAS-NETO *et al*., 2007b) e (DIAS-NETO *et al*., 2008).

[2] A justificativa para o uso deste ano como limite inferior da busca é que a pesquisa focou em um primeiro momento em TTBMs que utilizam a UML (www.omg.org) como base para modelagem, e 1990 foi o ano em que a UML foi oficialmente lançada.

- **Julho a Agosto de 2009:** esta segunda execução teve como objetivo atualizar o corpo de conhecimento de TTBMs, identificando apenas técnicas publicadas a partir de 2006 (ano da primeira execução) até 2009. Ao total foram identificados 193 novos artigos, reduzidos para 75 artigos após a aplicação de critérios de inclusão/exclusão. Tais artigos foram analisados e, juntamente com os 202 artigos identificados na primeira execução, eles compõem o corpo de conhecimento de TTBMs construído neste trabalho.

A seguir será apresentado resumidamente o planejamento, execução e análise dos resultados deste estudo para as duas rodadas de execução do protocolo de revisão sistemática. O "Apêndice A" apresenta a lista de fontes e artigos identificados ao longo das duas execuções deste protocolo de revisão sistemática.

As TTBMs identificadas e selecionadas foram analisadas quantitativa e qualitativamente. É importante destacar que o objetivo não foi comparar os méritos das diferentes técnicas, mas apenas extrair as informações de acordo com o texto publicado pelos autores, sempre analisando tais técnicas em grupos.

2.3.1 Planejamento e Execução da Revisão Sistemática

Os passos que envolvem o planejamento de uma revisão sistemática são: definição do objetivo e questões de pesquisa, seleção das fontes de busca, definição das *strings* de busca e dos critérios de inclusão/exclusão dos artigos e definição das estratégias de classificação e informações a serem extraídas de cada artigo. Usando a abordagem descrita em (BIOLCHINI *et al*., 2005), esses passos estão sumarizados a seguir:

- **Objetivo**: caracterizar as TTBMs publicadas na literatura técnica.
- **Questão de pesquisa**: quais são as TTBMs publicadas na literatura técnica e quais são suas principais características?
- **Fontes de busca**: seis bibliotecas digitais (ACM Portal, Compendex IE, IEEEXplorer, INSPEC, SCOPUS, Web of Science), assim como *websites* e anais de conferências.
- **String de busca genérica (que precisou ser adaptada a cada biblioteca digital)**: (approach or method or methodology or technique) and (("model based test") or ("model based testing") or ("model driven test") or ("model driven testing") or ("specification based test") or ("specification based testing") or ("specification driven test") or ("specification driven testing") or ("use case based test") or ("use case based testing") or ("use case driven test") or ("use case driven testing") or ("uml based test") or ("uml based testing") or ("uml driven test") or ("uml driven

testing") or ("requirement based test") or ("requirement based testing") or ("requirement driven test") or ("requirement driven testing") or ("finite state machine based test") or ("finite state machine based testing") or ("finite state machine driven test") or ("finite state machine driven testing")) and (software).

Para classificação das TTBMs, foram adotadas cinco categorias que separam os artigos descrevendo técnicas que utilizam diagramas da UML das técnicas que não seguem esta notação, além dos artigos que descrevem as técnicas aplicadas a teste funcional das técnicas aplicadas a teste estrutural (Figura 2.2).

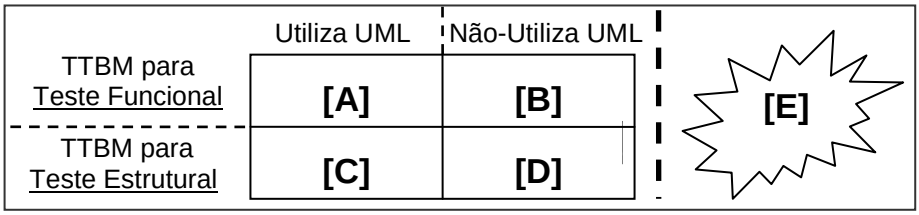


Figura 2.2. Categorização dos Artigos

[A] Possui um modelo representando informações provenientes dos requisitos do software (teste funcional) e o modelo é descrito usando diagramas UML.

[B] Possui um modelo representando informações provenientes dos requisitos do software (teste funcional) e o modelo é descrito em uma notação que não seja UML.

[C] Possui um modelo representando informações provenientes da estrutura interna do software (arquitetura, interfaces, unidades, teste estrutural) e o modelo é descrito usando diagramas UML.

[D] Possui um modelo representando informações provenientes da estrutura interna do software (arquitetura, interfaces, unidades, teste estrutural) e o modelo é descrito em uma notação que não seja UML.

[E] Artigos coletados durante a busca, mas que não descrevem Técnicas de Teste Baseado em Modelos e, portanto, devem ser excluídos.

Ao total, 599 publicações foram identificadas (Tabela 2.1), dos quais 328 artigos foram excluídos, pois seus escopos não estavam relacionados com este trabalho, eram repetidos ou não estavam disponíveis eletronicamente. Dos 271 artigos que sobraram, todos foram analisados e eles descrevem 219 TTBMs diferentes. Dessas, 76 TTBMs adotam diagramas UML e 143 TTBMs não utilizam a UML como notação para representação do software. A lista completa dos 599 artigos identificados ao longo das duas execuções do protocolo de revisão sistemática está disponível no Apêndice A deste trabalho.

Tabela 2.1. Classificação dos Artigos Identificados

Categorias de Artigos	Quantidade de Artigos	Percentual de Artigos
Categoria “A”	42	7,01%
Categoria “B”	116	19,36%
Categoria “C”	37	6,17%
Categoria “D”	76	12,68%
Categoria “E” ou Não Classificado	328	54,75%
Total de Artigos	**599**	**100%**

A divisão dos artigos selecionados (Categorias “A” a “D”) por ano de publicação está apresentada na Tabela 2.2, na qual é possível observar o aumento da quantidade de artigos descrevendo TTBMs nos últimos anos.

Tabela 2.2. Classificação dos Artigos Selecionados

Artigos por Ano de Publicação	Quantidade de Artigos	Percentual de Artigos
1990	1	0,37%
1991	1	0,37%
1992	2	0,73%
1993	3	1,13%
1994	5	1,84%
1995	1	0,37%
1996	11	4,06%
1997	5	1,84%
1998	7	2,58%
1999	16	5,90%
2000	13	4,79%
2001	11	4,06%
2002	12	4,42%
2003	17	6,27%
2004	25	9,22%
2005	31	11,44%
2006	37	13,65%
2007	32	11,80%
2008	35	12,91%
2009	6	2,21%
Total de Artigos	**271**	**100%**

Apos a identificação, seleção, classificação e extração das informações individuais de cada técnica, o próximo passo realizado foi analisá-las em conjunto.

2.3.2 Análise das Técnicas de Teste Baseado em Modelos

Os trabalhos identificados foram analisados de duas formas: quantitativa e qualitativamente.

2.3.2.1 Análise Quantitativa

Primeiramente foi realizada uma análise estatística a respeito de diversos resultados: por nível de teste, por características de qualidade de software avaliadas e por modelos usados pelas TTBMs.

- **Análise por Nível de Teste**

TTBMs foram caracterizadas baseando-se no nível de teste em que elas são aplicadas. Esta informação é importante para apresentar o escopo em que pesquisas relacionadas a TBM estão sendo desenvolvidas.

Testes podem ser aplicados para unidades pequenas do software (Teste de Unidade), coleção de unidades (Teste de Integração) ou o sistema inteiro (Teste de Sistema). TBM pode auxiliar nas atividades de teste em todos os níveis e em diferentes formas. Além disso, Teste de Regressão pode ser realizado com mais simplicidade, pois caso o modelo (ou especificação) mude, novos casos de teste podem ser gerados a partir do processo de geração automática. A Tabela 2.3 descreve os resultados da análise das TTBMs por nível de teste.

Tabela 2.3. Análise das TTBMs por Nível de Teste

Nível de Teste	# TTBMs
Sistema	155
Integração	34
Unidade	35
Regressão[3]	7

TBM foi originalmente projetada para Teste de Sistema (RAMAMOORTHY *et al*., 1976; CHOW, 1978), o que justifica o alto número de TTBMs aplicadas a este nível de teste (71%[4]). Em seguida, TTBMs são mais aplicadas a outros níveis, como Teste

[3] Teste de Regressão não seria a princípio um Nível de Teste, mas sim uma estratégia para re-execução dos testes independentemente do seu nível. No entanto, algumas técnicas foram projetadas específicamente para apoiar Testes de Regressão sem indicar para quais níveis de teste, e por isto foi criada uma categoria específica com essas TTBMs.

[4] O somatório das TTBMs apresentadas na Tabela 2.3 é um valor maior que 219, pois existem técnicas que possibilitam avaliar mais de um nível de teste.

de Unidade (16%), Integração (15%) e Regressão (3%). Teste de unidade não é ainda um nível de abstração tão usual em TBM, pois o objetivo deste nível de teste é testar pequenos módulos, funções ou classes após sua implementação. Existem diversas técnicas de apoio a teste estrutural a partir de código fonte (técnicas baseadas em código).

- **Análise por Ferramentas de Apoio**

As TTBMs foram ainda caracterizadas pela disponibilização de ferramenta para apoiar o processo de geração de casos de teste. Ferramentas podem reduzir o custo e tempo na aplicação das TTBMs, pois alguns passos podem ser automatizados, simplificando o uso da técnica. A Tabela 2.4 descreve o resultado da análise por ferramentas de apoio, na qual foi observado que quase 70% das TTBMs identificadas indicam a existência de apoio ferramental. Para as demais, não foi possível identificar a existência de ferramental de apoio, pois não houve registro nos artigos identificados na revisão sistemática.

Tabela 2.4. Análise das TTBMs por Nível de Teste

Usam Ferramenta de Apoio	Não citaram
153 (69,86%)	66 (30,14%)

- **Análise por Requisitos Não-Funcionais**

Requisitos de software podem ser funcionais ou não-funcionais. Requisitos funcionais definem cenários e funcionalidades que o software deve prover durante sua execução. Requisitos não-funcionais (RNF) são essenciais para definir aspectos arquiteturais e restrições durante o desenvolvimento do software. Ambos necessitam serem testados. A geração de casos de teste a partir da descrição de requisitos não-funcionais é em algumas situações não realizada. A seguir é descrita uma análise de TTBMs que possibilitam avaliar tais tipos de requisitos.

As categorias de RNF utilizadas nesta análise foram definidas usando o padrão ISO 9126 (2001). Este padrão define atributos de qualidade para software em 6 grupos (*Funcionalidade*, *Confiabilidade*, *Usabilidade*, *Manutenibilidade*, *Portabilidade* e *Eficiência*). No entanto, *Funcionalidade* é composta por outras sub-categorias (incluindo requisitos funcionais). Então, somente a sub-categoria *Segurança* relacionada ao grupo *Funcionalidade* foi usada para classificar requisitos não-funcionais neste trabalho. Sendo assim, o conjunto de categorias de RNF utilizado neste trabalho é composto por: eficiência, confiabilidade, usabilidade, manutenibilidade, portabilidade e segurança.

Tabela 2.5. Número de TTBMs que avaliam RNF de acordo com a ISO 9126 (2001)

Requisitos Não-Funcionais (de acordo com ISO 9126)	TOTAL
Eficiência	27
Usabilidade	6
Confiabilidade	2
Manutenibilidade	0
Portabilidade	0
Segurança	16

A partir da Tabela 2.5, pode ser observado que a maioria das TTBMs que possibilitam a avaliação de RNF é aplicada à avaliação da *eficiência*. Ao total, 27 TTBMs utilizam descrição sobre eficiência nos modelos para prover a geração dos testes. Tais TTBMs em sua maioria são destinadas à avaliação do tempo de resposta (desempenho) do software durante sua execução. Além de eficiência, 16 TTBMs utilizam requisitos de *segurança* durante a geração dos testes, 6 são aplicadas à avaliação de *usabilidade* e 2 avaliam *confiabilidade*.

- **Análise por Modelos usados para descrever a Estrutura/Comportamento do Software**

O modelo usado para descrever a estrutura ou comportamento do software é um elemento importante no estudo de TBM. O modelo define a limitação de cada técnica baseando-se na informação que ele pode representar sobre o software. Em algumas situações o modelo usado para um domínio de aplicação específico e não se adéqua a outro domínio. A Tabela 2.6 lista os modelos utilizados pelas TTBMs, classificados como modelos UML ou modelos Não-UML.

Tabela 2.6. Número de TTBMs por Modelo

Modelos UML	Modelos Não-UML
76 (34,70%)	143 (65,30%)

Entre os modelos UML, os mais adotados por TTBMs, nessa ordem, são: Diagrama de Estado, Diagrama de Classes e Diagrama de Sequência, seguidos pelos demais diagramas UML. Entre os modelos Não-UML, os mais adotados são: Máquina de Estado Finito, Especificação na Linguagem Z, Grafos, Cadeias de Markov e Modelos em XML.

Os dados coletados a partir desta revisão sistemática permitem ser observados sobre outras diferentes perspectivas (ex: TTBMs por plataforma de execução do software, grau de complexidade das TTBMs, quais ferramentas utilizam modelos intermediários). Estas informações podem apresentar cenários em Engenharia de Software não cobertos pelas soluções atuais observadas nas TTBMs.

2.3.2.2 Resumo das TTBMs identificadas

A Tabela 2.7 apresenta um resumo da caracterização das 219 TTBMs selecionadas para avaliação. Todas as TTBMs possuem ao menos um passo não-automatizado, porém este pode possuir diferente grau de complexidade. Sendo assim, a coluna "nível de complexidade" está representada em ícones que significam *Baixo*, *Médio* ou *Alto*, conforme a explicação a seguir:

- **Baixo (✔):** composta basicamente por passos automatizados ou passos manuais cuja tarefa da equipe de teste é apenas a de selecionar uma opção (ex: um dos critérios para geração dos testes ou quais casos de testes serão executados) disponibilizada pela TTBM;
- **Médio (⚠):** composta basicamente por passos manuais que requerem um maior esforço, tal como a modelagem/geração de modelos intermediários ou definição dos dados dos testes. Muitas TTBMs usam modelos intermediários durante o processo de geração dos testes, mas na maioria dos casos eles são gerados automaticamente (complexidade baixa). Quando esta geração é feita manualmente, o esforço será de complexidade média;
- **Alto (✘):** composta basicamente por passos manuais que possuem alta complexidade, tais como transformações manuais entre modelos, pois isso envolve a aplicação de regras de transformação e restrições para derivar um modelo, o que pode tornar o processo de geração dos testes uma tarefa bastante custosa ao projeto.

Tabela 2.7. Caracterização das Técnicas de Teste Baseado em Modelos

Autor(es)	Cat	Nível de Teste	Categoria de Software	Modelos Comportamentais e Estruturais	Ferramenta	Complexidade dos passos	Teste de RNF[5]
Abdurazik and Offutt, 2000	C	Sistema	OO	Diagrama de Colaboração	ND	✓	Não
Aichernig and Delgado, 2006	B	Unidade	Sistema Concorrente	Especificações em Input-Output Labelled Transaction System	Sim	✗	Não
Al-Amayreh and Zin, 1999	B	Sistema	Não definido	Especificação Z e PROLOG	Sim	✗	Não
Alagar *et al*., 2000	B	Unidade	Sistema Reativo e de Tempo Real	Especificação de Configuração do Sistema e Grid Automaton	Sim	✗	Eficiência
Alava *et al*., 2006	D	Integração	Aplicação Web	Design View (DView) e Arquivo XML	Sim	✗	Não
Ali *et al*., 2005	C	Integração	OO	Diagrama de estado, colaboração e SCOTEM (State COllaboration TEst Model)	Sim	✓	Não
Ambrosio *et al*., 2007	D	Sistema	Sistema Embarcado	Modelo baseado em Estado	Sim	⚠	Eficiência
Amla e Ammann, 1992	B	Unidade	Não definido	Especificação Z	ND	⚠	Não
Ammann and Offutt, 1994	B	Sistema	Não definido	Especificação Z	ND	✓	Não
Andrews *et al*., 2005	B	Sistema	Aplicação Web	Máquina de Estado Finita	Sim	✗	Não
Auguston *et al*., 2005	B	Sistema	Sistema Reativo e de Tempo Real	Modelos AEG (Attributed Event Grammar)	Sim	✗	Não
Avritzer e Weyuker, 1994	B	Sistema	Sistema de telecomunicação	Cadeia de Markov	Sim	✗	Eficiência
Baharom e Shukur, 2008	D	Unidade	Não definido	Especificação de Interface e Documento de Projeto Interno do Módulo	Sim	✗	Não
Bai *et al*., 2008	A	Sistema	Web Service	TOL (Test Ontology Language)	Sim	⚠	Não
Barbey *et al*., 1996	B	Unidade	OO	CO-OPN/2	Sim	✓	Não
Basanieri and Bertolino,2000	C	Integração	OO	Diagrama de Casos de Uso, Classes e Seqüência	ND	✗	Não
Bellettini *et al*., 2005	C	Sistema	Aplicação Web	Classes estendidas e Diagramas de Estado	Sim	✓	Não
Belli e Guldali, 2005	B	Sistema	Não definido	Máquina de Estado Finito, Lógica Temporal Linear	Sim	✗	Não
Benz, 2007	B	Integração	Sistema Distribuído	Modelo de Tarefas	Sim	✗	Não
Bernard *et al*., 2006	A	Sistema	Não definido	Diagramas de Classe, Objeto, Estado e Objects Constraint Language (OCL)	Sim	✓	Não
Bertolino *et al*.,2000	D	Integração	Não definido	Modelo Dinâmico LTS	Sim	✗	Não
Bertolino e Gnesi,2003	B	Sistema	Linha de Produto	Casos de Uso	Sim	✗	Não
Bertolino *et al*. ,2003	C	Integração	Software baseado em Componente	Diagrama de Caos de Uso, Seqüência e Classes	Sim	✓	Não
Bertolino *et al*., 2005	C	Integração	OO	Diagramas de Seqüência e Estado, Modelo Completo Racional	Sim	✗	Não
Biberstein e Fitzgerald, 1999	B	Sistema	Sistema de Tempo Real	Modelos DORIS/ADL	Sim	✗	Eficiência
Bigot et al., 2006	C	Sistema	SIM Smart Card	Diagrama de Estados UML	Sim	⚠	Não
Blackburn e Busser. 1996	B	Sistema	Sistema Crítico	Modelo de Especificação T-VEC	Sim	⚠	Não
Boden e Busser, 2004	D	Sistema, Unidade e Integração	Sistema Embarcado	Modelos Simulink	Sim	✗	Não
Bogdanov e Holcombe, 2001	D	Sistema	Sistema Embarcado, Sistema de Controle de Avião	Diagrama de Estado	Sim	✗	Não
Bonifacio *et al*., 2008	D	Unidade	Não definido	Máquina de Estado Finito	ND	✗	Não
Botaschanjan *et al*., 2004	A	Sistema	OO	Diagramas de Objeto e Seqüência e OCL	ND	⚠	Não
Botaschanjan *et al*., 2008	D	Sistema	Sistema Automotivo e Crítico de Segurança	Modelo de Tarefa AutoFOCUS	Sim	✗	Eficiência
Bouquet *et al*., 2005	B	Sistema	Software de Smart Card e Software de Real-Life	Máquina Abstrata na Linguagem B	Sim	⚠	Não
Bouquet *et al*., 2006	B	Unidade	OO	Especificação em JML	Sim	✓	Não

[5] RNF = Requisitos Não-Funcionais.

Autor(es)	Cat	Nível de Teste	Categoria de Software	Modelos Comportamentais e Estruturais	Ferramenta	Complexidade dos passos	Teste de RNF[5]
Bouquet *et al*., 2007	A	Sistema	OO	Diagramas de Classes, Objeto e Estado da UML, Expressões OCL	Sim	✓	Não
du Bousquet *et al*., 1999	D	Sistema	Software Reativo	Descrição do Ambiente	Sim	✓	Segurança
Boyd e Ural, 1991	B	Sistema	Não definido	Máquina de Estado Finito	ND	✓	Não
Briand and Labiche, 2002	A	Sistema	OO	Diagrama de Casos de Uso + Atividade, Seqüência, Colaboração, Classes, Dicionário de Dados (OCL)	Sim	✓	Não
Briand *et al*., 2002	A	Regressão	OO	Diagramas de Classes, Casos de Uso e Seqüência	Sim	✓	Não
Briand *et al*., 2002	C	Integração	OO	Diagrama de Classes e Grafo	ND	✓	Não
Briand *et al*., 2004	C	Sistema	OO	Diagrama de Estado + OCL, Classes, e Seqüência de Teste de Transação	Sim	✓	Não
Briand *et al*., 2004	C	Sistema	OO	Diagrama de Estado + OCL, Grafo de fluxo de evento/ação	ND	✗	Não
Briand *et al*., 2006	C	Unidade	COTS	Diagramas de Classes, Seqüência + OCL, restrições CSPE (Constraints on Succeeding and Preceding Events) e grafo	Sim	⚠	Não
Brinksma, 2004	B	Sistema	Sistema Reativo e de Tempo Real	VIZ	Sim	✗	Eficiência
Brito *et al*., 2009	C	Integração e Unidade	Sistema Embarcado	Diagramas de Colaboração e Seqüência da UML	Sim	✓	Eficiência
Buchs *et al*., 2006	A	Sistema	Sistema Concorrente OO	Modelo de Ambiente, Conceitual, de Protocolo e de Operação, CO-OPN (Concurrent Object Oriented Petri Nets)	Sim	✗	Não
Calame *et al*., 2007	A	Sistema	OO	Modelos UML 2.0	ND	✓	Não
Carpenter, 1999	A	Sistema	Sistema Crítico de Segurança	Diagramas de Caso de Uso e Seqüência	ND	✗	Não
Cavarra *et al*., 2003	A	Sistema	OO	Diagramas de Classes, Estado, Objeto e Formato Intermediário (ASM - Abstract State Machine)	Sim	✓	Não
Chang *et al*., 1990	D	Sistema	Sistema Distribuído	Redes de Petri	Sim	✗	Eficiência
Chang and Richardson, 1999	D	Unidade	Não definido	Especificação ADL e Descrição de Dados de Teste (TDD)	Sim	⚠	Não
Chang *et al*., 1996	D	Unidade	Não definido	Especificação ADL	Sim	⚠	Não
Chen *et al*., 2005	A	Sistema	OO	Diagrama de Atividades da UML	Sim	✓	Não
Chen e Liu, 2004	D	Sistema	Não definido	Especificações em SOFL	ND	✗	Não
Chen *et al*., 2002	A	Regressão	OO	Diagrama de Atividades	ND	✓	Não
Chen *et al*., 2005	D	Integração	Não definido	Diagrama de Fluxo de Dados de Condição em SOFL (Structured Object-Oriented Formal Language)	ND	⚠	Não
Chetali e Nguyen, 2009	D	Unidade	Smart Card	SyncCharts, Máquina da Estado Finito	Sim	✗	Não
Chevalley and Fosse, 2001	A	Sistema	Software Crítico	Diagrama de Estado	Sim	✓	Não
Choi, 2001	B	Sistema	OO	Caso de Uso Estendido	Sim	⚠	Não
Chung *et al*., 1999	B	Sistema	Sistema Concorrente	Especificação do sistema escrita em MSC	Sim	✗	Não
Chung *et al*., 1996	D	Unidade	Não definido	Máquina de Estado Básico, Máquina de Estado Composto	ND	✓	Não
Conrad e Krupp, 2006	B	Sistema	Sistema Embarcado	Árvore de Classificação	Sim	✗	Não
Crichton *et al*., 2001	C	Sistema	OO	Modelo do Sistema (Diagramas de classes, objeto e estado) e Diretivas de Teste (Diagramas de Objeto e Estado), Formato Intermediário	ND	✓	Não
Dai-Zhe et al., 2004	C	Integração	Sistema Embarcado	Diagramas de Interação, Atividade e Estado da UML	ND	✗	Eficiência
Dalal *et al*., 1999	B	Sistema	Não definido	Requisitos descritos em AETGSpec	Sim	✓	Não
Dan e Aichernig, 2005	D	Sistema	Não definido	Especificação em RSL	Sim	✗	Não
Darmaillacq et al., 2008	D	Sistema	Não definido	Modelo de Falha	ND	✗	Segurança
Deng *et al*., 2004	A	Sistema e Regressão	OO	Diagramas de Casos de Uso, Classes, Estado Seqüência, Colaboração e Atividade	Sim	⚠	Não
Dias e Vieira, 2000	D	Integração	OO	Diagrama de Estado	Sim	⚠	Não
Donat, 1997	B	Sistema	Não definido	Especificação de Requisitos	Sim	✗	Não
Dulz e Zhen, 2003	C	Integração	Não definido	Diagrama de Seqüência Anotada e MCUM (Markov Chain Usage Model)	Sim	✓	Não

Autor(es)	Cat	Nível de Teste	Categoria de Software	Modelos Comportamentais e Estruturais	Ferramenta	Complexidade dos passos	Teste de RNF[5]
Edwards, 2000	B	Integração	OO	Gráfico de Fluxo de Controle	ND	✗	Não
El-Fakih et al., 2008	D	Sistema	Não definido	Máquina de Estado Finito Estendida	ND	⚠	Não
Ernits et al., 2006	D	Sistema	Não definido	Modelos UPPAL	Sim	⚠	Não
Fraser e Wotawa, 2007	D	Unidade	Não definido	Modelos NuSMV	Sim	✓	Não
Friedman *et al*., 2002	B	Sistema	Software Concorrente	Máquina de Estado Finita	Sim	✓	Não
Gallagher e Offutt, 2006	D	Integração	OO	Grafo de Fluxo de Componente	Sim	✓	Não
Gargantini, 2007	D	Sistema	Não definido	Máquina de Estado Abstrato	Sim	✗	Não
Gargantini and Heitmeyer,1999	B	Sistema	Não definido	Especificação SCR (Software Cost Reduction)	Sim	✓	Não
Gargantini et al.,, 2009	A	Sistema	Sistema Embarcado	Máquina de Estado Abstrato	Sim	✗	Eficiência
Garousi *et al*., 2006	C	Sistema (Teste de Stress)	Sistema Distribuído	Diagramas de Classes, Seqüência, Contexto, Implantação de Rede e de Visão Geral de Interação Modificada, Modelo de Fluxo de Controle, Árvore de Interconectividade de Rede, Padrão de Utilização de Tráfico de Rede, e restrições inter-SD	ND	⚠	Eficiência e Confiabilidade
Gnesi *et al*., 2004	C	Sistema	OO	Diagrama de Estado e IOLTS (transitions labeled by input/output-pairs)	ND	✓	Não
Gross *et al*., 2005	C	Integração	OO	U2TP (UML 2.0 Testing Profile) e TTCN-3 (Testing and Test Control Notation)	ND	✓	Não
Guo et al., 2005	B	Sistema	Não definido	Máquina de Estado Finito	ND	✗	Não
Gutierrez et al., 2008	A	Sistema	Não definido	Diagramas de Caso de Uso e Atividades da UML	Sim	✓	Não
Hagar e Bieman, 1996	B	Sistema	Sistema Crítico de Segurança	Especificação Anna	Sim	✗	Não
Hanna e Munro, 2007	B	Sistema	Web Service	Modelo de Abstração Formal do Esquema XML	Sim	✓	Não
Hartman and Nagin, 2004	A	Sistema	Sistema Distribuído	Combinação de diagramas de estado de classes e objeto, diagrama de estado e arquivo XML	Sim	✓	Não
Hartmann *et al*., 2000	C	Integração e Unidade	OO	Diagrama de Estado e Grafo	Sim	✓	Não
Hasling et al, 2008	A	Sistema	Sistema Médico OO	Diagramas de Caso de Uso, Classes, Atividade, Pacote e Sequência da UML	Sim	✓	Não
Hayes e Sankar, 1994	D	Unidade	Sistema Embarcado	Modelo em TDD (Test Data Description)	ND	✓	Não
Hessel e Pettersson, 2007	B	Sistema	Sistema Embarcado	Rede de Comunicação em Máquina de Estado Finito Estendida	Sim	✓	Não
Hierons et al., 2009	B	Sistema	Não definido	Modelo de falha em Máquina de Estado Finito Estocástica	ND	✓	Eficiência
Hong *et al*., 2000	B	Sistema	Não definido	Diagrama de Estado e Máquina de Estado Finito Estendida	Sim	✓	Não
Howe et al., 1997	B	Sistema	Sistema Embarcado	Especificação em LISP	Sim	✗	Não
Hsia et al., 1997	B	Sistema	Não definido	Máquina de Estado Finito baseada em cenário, Árvore de Cenário	Sim	✗	Não
Hungar et al., 2003	B	Sistema	Sistema de Telecomunicação	Automato Finito	Sim	✗	Não
Ipate e Holcombe, 2005	B	Unidade	OO	Máquinas X stream, Máquina de Estado Finito	ND	✗	Não
Jategaonkar et al., 1998	B	Sistema	Sistema Reativo	Máquina de Estado Finito	Sim	⚠	Segurança
Jeannet et al., 2007	B	Sistema	Sistema Reativo	Especificações em Input/Output Symbolic Transition Systems (ioSTS)	Sim	✗	Não
Jia et al., 2008	A	Sistema	Aplicação Web	Diagrama de Atividade da UML	Sim	✓	Usabilidade
Jia, 1993	B	Sistema	Não definido	Tipos de Dados Abstratos (ADT)	ND	✓	Não
Jourdan et al., 2006	B	Sistema	Estações Sun usando Solaris Sparc 5.8	Máquina de Estado Finito Estendida	Sim	✓	Não
Jurjens,2008	A	Sistema	Sistema Crítico de Segurança	Modelos UMLsec	Sim	✓	Segurança
Kahsai et al., 2008	B	Sistema	Linha de Produto	Especificação em CSP-CASL	Sim	✗	Não
Kandl et al., 2007	B	Sistema	Sistema Embarcado	Modelo NuSMV	Sim	✗	Segurança
Kansomkeat and Rivepiboon, 2003	A	Sistema	OO	Diagrama de Estado e Grafo de Fluxo de Teste	ND	✓	Não
Kaplan et al., 2008	A	Sistema	OO	Diagramas de Classe, Casos de Uso e Objeto da UML	Sim	✓	Não
Katara e Kervinen, 2007	B	Sistema	Sistema Embarcado	Caso de Uso e LTS	ND	✓	Não

Autor(es)	Cat	Nível de Teste	Categoria de Software	Modelos Comportamentais e Estruturais	Ferramenta	Complexidade dos passos	Teste de RNF[5]
Kervinen et al., 2005	B	Sistema	Sistema Embarcado	Modelo de Teste	Sim	✗	Não
Kim *et al*., 1999	C	Unidade	OO	Diagrama de Estado e Máquina de Estado Finito Estendida	ND	✓	Não
Kissoum e Sahnoun, 2007	D	Unidade	Sistema Distribuído	Diagrama de Requisitos InterElement, Rede de Petri	Sim	✗	Não
Kissoum e Sahnoun, 2008	C	Unidade	Sistema Distribuído	Diagrama de Sequência da UML	Sim	✗	Não
Koo e Mishra, 2008	D	Unidade	Sistema Embarcado	Máquina de Estado Finito	ND	✗	Não
Koopman et al., 2007	B	Sistema	Aplicação Web	Máquina de Estado Finito Estendida	Sim	✗	Usabilidade
Koopman e Plasmeijer, 2006	B	Sistema	Sistema Reativo	Máquina de Estado Finito Estendida	Sim	✗	Não
Korel et al., 2002	D	Regressão	Não definido	Máquina de Estado Finito Estendida	ND	⚠	Não
Korel et al., 2007	D	Unidade	Não definido	Máquina de Estado Finito Estendida	ND	✓	Não
Kwang e Eun, 2007	A	Sistema, Integração	OO	Diagramas de Casos de Uso, Classe, Sequência, OCL e MM-Path	ND	✓	Não
Lallali et al., 2008	B	Unidade	Web Service	Modelos BPEL	Sim	✗	Eficiência
Leathrum e Liburdy, 1996	B	Sistema	Não definido	Especificação em ADA	Sim	✗	Não
Legeard *et al*., 2004	D	Sistema	Software Crítico	Especificação Formal (B ou Z)	Sim	✓	Não
Leung e Wong, 2000	D	Sistema	OO	Grafo de Classes	ND	✓	Não
Li et al., 2007	B	Sistema	Sistema de Tempo Real	Formato Intermediário	Sim	✗	Segurança
Li et al., 2007	A	Sistema	Aplicação Web	Modelo de Navegação descrevendo validação de entradas do usuário	Sim	✓	Usabilidade
Liang, 2005	D	Regressão e Unidade	OO	Especificação em TCOZ	Sim	✓	Eficiência
Lihua et al., 2004	D	Regressão	Não definido	Máquina de Estado Finito e Especificação de Teste de regressão	Sim	✓	Não
Linzhang *et al*., 2004	A	Sistema	OO	Diagrama de Atividades	Sim	✓	Não
Liu et al., 2001	D	Sistema	Aplicação Web	Modelo de Teste de Aplicação Web e Diagrama de Relação de Objeto	ND	⚠	Não
Liu et al., 2002	D	Unidade	OO	Máquina de Estado Finito	Sim	✗	Não
Liuying and Zhichang, 1999	A	Sistema	OO	Diagrama de Estado e Máquina de Estado Finito	Sim	✓	Não
Lucio *et al*., 2005	A	Sistema	OO	Diagrama de Classes (Fondue), Diagramas de Colaboração e Estado UML, e operações OCL	ND	⚠	Não
Lund and Stølen, 2006	A	Sistema	Não definido	Diagrama de Seqüência	ND	✗	Não
Mandrioli *et al*., 1995	B	Sistema	Sistema Crítico e de Tempo Real	Especificação do Software em TRIO (Tempo Reale ImplicitO)	Sim	✓	Eficiência
Massicotte et al., 2007	C	Integração	Orientado a Aspecto	Diagrama de Colaboração da UML	Sim	✓	Não
Massink et al., 2006	C	Sistema	Sistema Concorrente	Diagrama de Estado da UML	ND	✗	Não
Masson et al., 2007	B	Sistema	Smart Card	Model Funcional Formal do Sistema	Sim	✗	Segurança
Memon, 2007	B	Sistema	Sistema de Informação	Modelo de Fluxo-Evento	Sim	✓	Usabilidade
Memon et al., 2000	B	Sistema	OO	Especificação de GUI	Sim	✗	Não
Merayo e Nunez, 2008	B	Sistema	Sistema de Tempo Real	Extensão de Máquina de Estado Finito	ND	✗	Eficiência
Meyer and Sandfoss, 1998	A	Sistema (GUI)	Não definido	Caos de Uso + Perfil Operacional – Modelo de Engenharia de Teste	Sim	✗	Não
Mikucionis et al., 2004	B	Sistema	Sistema Embarcado e de Tempo Real	Automato Finito	Sim	✗	Eficiência
Mingsong *et al*., 2006	A	Sistema	Programa em Java	Diagrama de Atividades	Sim	⚠	Não
Misailovic et al., 2007	D	Sistema	Não definido	Gráfico Acíclico Direcionado	Sim	✓	Não
Morasca et al., 1996	B	Sistema e Integração	Sistema de Tempo Crítico	Especificação do Módulo na linguagem TRIO+	ND	✗	Eficiência
Murphy et al., 2009	B	Sistema	OO	Modelo em JML	Sim	✓	Não
Murray *et al*., 1998	D	Unidade	OO	Máquina de Estado Finito e Grafo de Teste	Sim	✗	Não
Murthy *et al*., 2006	A	Sistema	Não definido	Diagrama de Estado Estendido, CTGM Estendido	Sim	✓	Não
Nachmanson et al., 2004	D	Sistema	Sistema Não-dterminístico	Grafo Direcionado	Sim	⚠	Não
Naslavsky et al., 2007	A	Sistema	OO	Diagrama de Classes e Sequência da UML, Modelo de Fluxo de Controle e Modelo Hierárquico	Sim	✓	Não

Autor(es)	Cat	Nível de Teste	Categoria de Software	Modelos Comportamentais e Estruturais	Ferramenta	Complexidade dos passos	Teste de RNF[5]
Nebut and Fleurey, 2006	A	Sistema	Software Embarcado OO	Diagrama de Casos de Uso e Seqüência, e Modelo de Simulação	Sim	⚠	Não
Nilson et al., 2004	D	Sistema	Sistema Concorrente e de Tempo Real	Automato Temporal com modelo de Tarefas	Sim	✗	Eficiência
Offutt and Liu, 1999	B	Sistema e Unidade	Software Crítico	Condition data flow diagram (CDFD) em SOFL (Structured Object-Oriented Formal Language)	ND	✓	Não
Offutt and Abdurazik, 1999	A	Sistema	OO	Diagrama de Estado	Sim	✓	Eficiência
Offutt et al., 2003	A	Sistema	OO	Diagrama de Estado e Grafo de Especificação	Sim	✗	Não
Okika et al., 2006	D	Sistema	Sistema Embarcado	Modelos em *Testing and Test Control Notation* (TTCN-3)	Sim	✗	Não
Olimpiew and Gomaa, 2005	A	Sistema	Linha de Produto de Software	Modelo de Funcionalidades, Modelo de Casos de Uso, Modelos estáticos (classes) e dinâmicos (estado e interação de objeto), modelo arquitetural de software baseado em componentes	ND	✓	Não
Paiva et al., 2005	B	Sistema	Sistemas em .NET	Máquina de Estado Finito	Sim	✗	Não
Paradkar, 2004	B	Sistema	Sistema Interativo	Automato de Estado Finito	ND	✗	Não
Paradkar, 2004	D	Sistema	Sistema Reativo	Modelo Operacional (SALT – Specification and Abstraction Language for Testing) e Máquina de Estado Finito Estendida	Sim	✓	Não
Paradkar et al., 1997	D	Sistema	Sistema de Tempo Real	Grafo de Causa-Efeito	ND	✗	Não
Paradkar et al., 2007	B	Sistema	Web Service	Modelos IOPEs	Sim	✓	Não
Pari Salas et al., 2007	B	Sistema	Não definido	Descrição do Comportamento do Sistema/Componentes e Modelo de Ataque	Sim	✗	Segurança
Parissis and Ouabdesselam,1996	B	Sistema	Sistema Reativo	Especificação do Ambiente e Propriedades de Segurança	Sim	✓	Segurança
Parissis e Vassy, 2001	B	Sistema	Sistema Síncrono e Reativo	Máquina de Estado de Simulação	Sim	✗	Segurança
Peraire et al., 1998	D	Unidade	Sistema Embarcado OO	Especificação em CO-OPN/2	Sim	✗	Não
Popovic e Velikic, 2005	B	Sistema	Sistemas em Java e C++	Não definido	Sim	✗	Não
Popovic e Kovacevic, 2007	B	Sistema	Sistema Reativo	Modelo de Perfil Operacional de Stress	Sim	✗	Não
Poston, 1994	B	Sistema	OO	Modelo Funcional OMT	Sim	✓	Não
Pretschner *et al*., 2008	B	Sistema	Não definido	Política de Acesso	ND	✓	Segurança
Pretschner *et al*., 2001	B	Sistema	Sistema Reativo e Embarcado	Diagramas de Estrutura do Sistema, Transição de Estado e de Tipo de Dados, e Gráficos de Seqüência de Mensagem	Sim	✓	Eficiência
Prowell, 2003	D	Integração	Sistemas em C	Modelo de Utilização do Sistema em Cadeia de Markov	Sim	✗	Não
Rajappa et al., 2008	B	Sistema	Sistema de Tempo Real	Grafo Direcionado	ND	✓	Não
Reuys et al., 2005	A	Sistema	OO	Diagramas de Atividade, Casos de Uso e Sequência da UML	Sim	✓	Não
Reza et al., 2008	A	Sistema	Aplicação Web	Diagrama de Estado da UML	Sim	✓	Usabilidade
Richardson *et al*., 1992	D	Sistema	Sistema Reativo	No estudo de caso, RTIL (Real Time Interval Logic) e Especificação Z	ND	⚠	Eficiência e Segurança
Richardson and Wolf, 1996	D	Integração	Não definido	CHAM (CHemical Abstract Machine)	ND	✓	Não
Riebisch *et al*., 2002	A	Sistema	OO	Diagramas de Casos de Uso e Estado, Grafo e Modelo de Utilização	Sim	✓	Não
Robinson-Mallett et al., 2008	B	Integração	Sistema Distribuído de Tempo Real	Modelos de Teste em Cadeias de markov	Sim	✗	Não
Rocha e Martins, 2008	C	Integração	COTS	Diagrama de Atividade UML	Sim	✓	Não
Rumpe, 2003	A	Sistema	OO	Diagrama de Objeto e Seqüência, OCL	ND	✗	Eficiência
Rutherford e Wolf, 2003	D	Unidade e Integração	Sistema Distribuído OO	Descrição do Sistema em XML	Sim	✓	Não
Sakurai et al., 2008	B	Sistema	Sistema Crítico de Segurança	Modelo Formal Customizável	Sim	✗	Eficiência
Santos-Neto et al., 2008	A	Sistema	Sistema de Informação OO	Diagrama de Classes e Estado da UML	Sim	✓	Eficiência e Usabilidade
Satpathy *et al*., 2007	B	Sistema	Não definido	Especificação em Linguagem B	ND	✗	Não
Satpathy *et al*., 2005	D	Sistema	OO	Grafo de Cobertura de Estado (B) e Especificação em PROLOG	Sim	✓	Não
Satpathy *et al*., 2008	B	Sistema	Sistema Embarcado	Grafo de Fluxo de Estado	Sim	✗	Não
Scheetz *et al*., 1999	C	Integração	OO	Diagrama de Classes e Estado + OCL	Sim	⚠	Não

Autor(es)	Cat	Nível de Teste	Categoria de Software	Modelos Comportamentais e Estruturais	Ferramenta	Complexidade dos passos	Teste de RNF[5]
Schroeder et al., 2003	B	Sistema	Não definido	Grafo Direcionado, Modelos de Dados	Sim	✗	Não
Seifert, 2008	C	Sistema	Sistema Embarcado	Diagrama de Estado da UML	Sim	✓	Não
Shu e Lee, 2007	B	Sistema	Sistema Crítico de Segurança	Máquina de Estado Finito Estendida	Sim	⚠	Segurança
Sinha and Paradkar, 2006	D	Integração	Web service	WSDL-S e Máquina de Estado Finito Estendida	ND	✓	Não
Sinha e Smidts, 2006	B	Sistema	Não definido	Especificações em HaskellDB, Máquina de Estado Finito Estendida	Sim	✗	Não
Sokenou, 2006	C	Integração e Unidade	OO	Diagrama de Seqüência, Estado de Protocolo e OCL	ND	✓	Não
Song et al., 2008	B	Sistema	Aplicação Web	Modelo Navegacional On-the-fly e Máquina de Estado Finito	ND	✓	Segurança
Stobie, 2005	B	Sistema	Qualquer categoria	Máquina de Estado Finito e Linguagem de Máquina de Estado Abstrato	Sim	✓	Não
Stocks and Carrington, 1996	D	Unidade	Não definido	Template de Teste, descrevendo entradas e resultados válidos	Sim	ND	Não
Tahat *et al*., 2001	B	Sistema e Regressão	Sistema Distribuído e Embarcado	Requisitos expressos em formato Textual e SDL (Specification Description Language), Máquina de Estado Finito Estendida	ND	✓	Não
Tan *et al*., 2004	B	Sistema	Não definido	Especificação de Software em LTL (linear temporal logic)	Sim	✓	Segurança e Eficiência
Tomita e Sakamura, 1999	D	Integração	Sistema Operacional	Máquina de Estado Finito Determinístico	Sim	✗	Não
Traore, 2003	A	Sistema	OO	Diagramas de Estado, Classes e Seqüência	Sim	⚠	Não
Vaysburg et al., 2002	D	Sistema	Não definido	Máquina de Estado Finito Estendida, Grafo de Dependência Estático e Dinâmico	ND	✗	Não
Vieira *et al*., 2006	A	Sistema (GUI)	Não definido	Diagramas de Casos de Uso + Atividade e Classes	Sim	⚠	Não
Vilkomir e Bowen, 2006	D	Sistema	Sistema Crítico de Segurança	Especificação em Notação Z	ND	✗	Não
Voigt et al., 2007	C	Sistema	Não definido	Diagrama de Estado da UML	ND	⚠	Não
Wang e Huang, 2008	D	Unidade	Web Service	Modelo de Requisitos OWL-S	Sim	✓	Não
Watanabe e Sakamura, 1996	D	Integração	Sistema de Tempo Real	Especificação em Notação Z usando Máquina de Estado Finito	ND	✗	Não
Weber et al., 1994	B	Sistema	Sistema Eletrônico de Instrumentação de Vôo	Modelo Dinâmico e Estático	ND	✗	Não
Whalen et al., 2006	B	Sistema	Sistema Embarcado	Propriedades LTL	ND	✗	Não
Wimmel e Jürjens, 2002	B	Sistema	Sistema Concorrente, Reativo e de Segurança Crítica	Lógica Proposicional, Diagrama de Estrutura, Modelo de Sistema AutoFocus	Sim	✗	Segurança e Eficiência
Wu *et al*., 2003	C	Integração	Software baseado em Componente	Diagrama de Colaboração/Seqüência ou Estado UML	ND	✓	Não
Xie et al., 2006	D	Unidade	OO	Máquina Abstrata de Estado de Objeto	Sim	✗	Não
Xu and Xu, 2006	D	Integração	Programas Orientados a Aspectos	Modelo de Estado (Aspect-oriented programming)	ND	✓	Não
Yan et al., 2004	A	Sistema	Sistema Distribuído e Crítico de Segurança	Diagramas de Casos de Uso, Sequência, Expressões, Cadeia de Markov	ND	✓	Confiabilidade
Yao e Wang, 2004	B	Sistema	Não definido	Especificações RPTA	ND	✗	Não
Yu et al., 2003	B	Sistema	Não definido	Árvore de Classificação	Sim	✗	Não
Zander et al., 2005	C	Sistema	Sistema Distribuído	UML 2.0 Testing Profile, Testing and Test Control Notation	Sim	✗	Não
Zheng *et al*., 2008	A	Unidade	OO	Diagramas de Interação, Classes da UML	Sim	✓	Não
Zhou et al., 2008	C	Integração	OO	Diagrama de Classes e Sequência da UML	ND	✓	Não

2.4 Considerações Finais do Capítulo

Este capítulo apresentou uma síntese da área de Teste Baseado em Modelos, apresentando as TTBMs identificadas na literatura e analisando os desafios para se prover a seleção de TTBMs para projetos de software. Este cenário introduz novos desafios para a área de Teste Baseado em Modelos, tais como:

- Quais são as TTBMs disponíveis na literatura técnica para serem aplicadas em projetos de software?
- Como identificar as principais características de uma TTBM antes de aplicá-la em projetos de software, identificando, por exemplo, os requisitos necessários para sua implantação?
- Como identificar qual contexto em que TTBMs podem ser aplicadas ou que possuem aplicação limitada? Em qual contexto elas são mais eficientes?
- Como fornecer informações úteis que auxiliem uma equipe na identificação de TTBMs mais adequadas a um determinado projeto de software?
- Como uma escolha inadequada a respeito da seleção de TTBMs afeta a qualidade dos testes de software em um projeto?
- Como apoiar a tomada de decisão a respeito da seleção de TTBMs para projetos e organizações de software? E qual o impacto da seleção combinada de mais de uma TTBM em variáveis do processo de testes de software (ex: esforço, custo ou cobertura)?
- Como minimizar os riscos associados à seleção de TTBMs? E qual o impacto dessa minimização na qualidade dos testes em projetos e organizações de software?

Grande parte das questões listadas acima está relacionada à construção de um corpo de conhecimento inicial sobre TTBMs, identificando e caracterizando tais técnicas, e possibilitando uma tomada de decisão baseada em conhecimento técnico e científico a respeito de quais TTBMs seriam mais adequadas a um projeto de software.

O Capítulo seguinte apresenta o corpo de conhecimento sobre TTBMs desenvolvido ao longo deste trabalho, e que faz parte da solução adotada para apoiar a seleção combinada de TTBMs em projetos de software.

CAPÍTULO 3 - CORPO DE CONHECIMENTO SOBRE TÉCNICAS DE TESTE BASEADO EM MODELOS

Neste capítulo será apresentado o corpo de conhecimento sobre Técnicas de Teste Baseado em Modelos desenvolvido a partir do resultado de estudos secundários e primários, sendo descrito seu conteúdo e estrutura.

3.1 Introdução

O primeiro passo visando ao desenvolvimento de uma abordagem de apoio à seleção de TTBMs consiste em conhecer quais TTBMs estão disponíveis e quais são suas principais características. Isso possibilitou a criação de um corpo de conhecimento sobre TTBMs. A existência de um corpo de conhecimento pode simplificar o processo de seleção de qual técnica é mais adequada para um projeto de software com certas características, pois irá prover um conhecimento necessário para apoiar esta tomada de decisão.

Neste trabalho, a construção do corpo de conhecimento sobre TTBMs foi realizada a partir dos resultados de dois estudos, um estudo secundário com o propósito de definição do conteúdo do corpo de conhecimento e um estudo primário com propósito de definição de sua estrutura.

As próximas subseções irão descrever o planejamento e resultados obtidos a partir destes dois estudos, viabilizando a definição do corpo de conhecimento.

3.2 Conteúdo do Corpo de Conhecimento

Com o propósito de se definir o conteúdo do corpo de conhecimento sobre TTBMs, foi planejada e executada uma revisão sistemática com o objetivo de identificar e caracterizar TTBMs publicadas na literatura técnica. Sendo assim, um protocolo de busca e análise foi adotado. Este protocolo, seus resultados e análises foram descritos no Capítulo 2 - Seção 2.3 deste trabalho. As TTBMs que compõem o corpo de conhecimento inicial sobre TTBMs são as 219 técnicas identificadas e caracterizadas na revisão sistemática descrita no Capítulo 2.

3.3 Estrutura do Corpo de Conhecimento: *Survey* com especialistas em TBM

Para a definição da estrutura do corpo de conhecimento sobre TTBMs, inicialmente foi realizada uma revisão informal da literatura, principalmente a partir de (DALAL *et al*., 1999), (PRETSCHNER *et al*., 2005), (VEGAS e BASILI, 2005) e (UTTING *et al*., 2006). Nesta revisão, 18 atributos divididos em 4 categorias. As categorias usadas para classificação foram definidas por VEGAS e BASILI (2005), e são:

- **Ferramentas:** uma TTBM pode ter ferramentas disponíveis ou requerer alguma ferramenta de apoio a algum de seus passos.
- **Histórico:** características referentes a resultados relacionados a cada TTBM obtidos anteriormente à seleção que está sendo realizada e que podem evoluir ao longo do tempo.
- **Objeto:** algumas características do software no qual as TTBMs serão aplicadas que podem determinar o uso de uma ou outra TTBM, por exemplo, o domínio do software, as características de qualidade requeridas, dentre outras.
- **Técnica:** algumas características de TBM que não são influenciadas pelo projeto de software onde estão sendo aplicadas, tais como critérios adotados ou modelos usados, e podem ser relevantes para decidir pelo uso de uma ou outra TTBM.

A Tabela 3.1 lista os atributos usados para caracterização de uma TTBM obtidos a partir da literatura técnica. Na coluna “Pertence ao EC” é indicado se o atributo pertence à abordagem de seleção de técnicas de teste chamada de esquema de caracterização proposta por VEGAS e BASILI (2005) ou não.

Tabela 3.1. Atributos utilizados para caracterizar uma TTBM

Categoria	Atributo	Pertence ao EC	Importância para TBM
Ferramenta	Ferramenta de Apoio	SIM	Define a existência ou não de uma ferramenta de apoio. Caso exista, é importante observar quais passos são apoiados pela ferramenta e quais são suas limitações.
	Necessidade de Ferramentas Externas	SIM	Define a necessidade do uso de alguma ferramenta externa à TTBM para viabilizar sua aplicação (ex: ferramenta para geração do modelo usado pela TTBM para geração dos testes).
Histórico	Avaliação Experimental	SIM	Indicador do tipo de avaliação experimental que a TTBM foi submetida.
	Resultados Históricos	SIM	Métricas que descrevem a utilização de uma TTBM em projetos anteriores em uma organização de software.

Objeto	Característica de Qualidade de Software que a técnica está apta a avaliar	SIM	Normalmente, TTBMs estão aptas a avaliar apenas um conjunto restrito de características de qualidade do software (ex.: usabilidade, desempenho, segurança, funcionalidade, eficiência, confiabilidade, controle de acesso, etc.).
	Domínio de Software	SIM	Define o escopo em que uma TTBM pode ser aplicada, tais como Plataforma de Execução ou Paradigma de Desenvolvimento – Sistemas Embarcados, Sistema cliente-servidor, Sistemas OO, Sistemas de tempo real, Aplicações Web, etc.
	Limitações/Restrições para usar uma técnica	NÃO	Essas informações dizem respeito ao que uma TTBM não está apta a fazer ou algum cenário onde ela não pode ser aplicada (ex: a técnica só pode ser aplicada em projetos com características específicas, tais como tamanho, habilidades, ciclo de vida). Isso pode restringir seu uso em projetos de software.
	Nível de Teste	SIM	Define qual nível de abstração de teste é usado pela TTBM para avaliar o software sob teste. Os níveis usados são: teste de sistema, integração e unidade.
	Tipo de Técnica de Teste	NÃO	Algumas TTBMs são aplicadas para teste funcional e outras para teste estrutural. É importante observar esta característica para apoiar a seleção de uma técnica para um projeto.
Técnica	Critério de Cobertura dos Testes	SIM	Define as regras usadas para gerar casos de teste a partir do modelo do software. Existem diversos tipos de critérios, como: fluxo de dados, fluxo de controle e análise de mutantes. Eles definem o esforço e qualidade dos resultados gerados automaticamente pela TTBM.
	Critério de Geração de Casos de Teste	SIM	Descreve os passos a serem seguidos pela TTBM desde a construção do modelo até a geração ou execução dos testes. Eles definem o que pode ser automatizado e o que não pode, e como integrar esses passos.
	Entradas requeridas para usar uma técnica	NÃO	TTBMs comumente requerem diferentes entradas (artefatos) para iniciar o processo de construção do modelo para geração dos casos de teste. Normalmente, essas entradas possuem diferentes complexidades para serem interpretadas. É importante observar se o processo de desenvolvimento de software está apto a produzir as entradas necessárias para usar a TTBM.
	Modelo Comportamental/ Estrutural	NÃO	Representa as características do software que podem ser testadas por uma TTBM. O modelo pode ser a maior limitação de uma técnica, pois indica que informações do software podem ser representadas ou não. Algumas vezes o modelo é usado para um domínio de aplicação específico e não pode ser usado em outro contexto. Três aspectos são importantes a respeito dos modelos: (1) quão fácil e automatizado é seu desenvolvimento, (2) se ele contém as informações necessárias para a geração dos testes, e (3) quão fácil é para extrair casos de teste a partir dele.
	Nível de Complexidade dos Passos Não-automatizados	SIM	Indicação a respeito do esforço, custo e tempo necessário para realizar os passos requeridos em uma TTBM. O grau de complexidade dos passos automatizados é normalmente baixo. No entanto, os passos não automatizados podem requerer diferente grau de esforço, custo e habilidade para ser realizado. Assim, torna-se necessário analisar seus graus de complexidade indicando mais precisamente o nível de automação de uma TTBM.
	Proporção de Passos Automatizados	SIM	Relação entre a quantidade de passos automatizados que compõem a TTBM e a quantidade de passos total. A principal característica de TBM é a possibilidade de geração e execução automática de casos de teste, pois isso pode resultar em menos tempo e esforço para o processo de testes.

	Resultados gerados pela técnica	SIM	TTBMs podem gerar resultados em diferentes formatos (ex: roteiros de teste a serem executados em plataformas específicas ou resultados dos testes sumarizados após a execução dos casos de teste). É importante observar se os resultados gerados estão de acordo com os artefatos planejados para o processo de desenvolvimento de software.
	Tecnologia de Geração dos Testes	NÃO	Diferentes TTBMs usam diferentes tecnologias para geração dos casos de testes (ex: UML, Linguagem B, Especificação Z, TSL e outros). É importante observar se o processo de desenvolvimento usa a mesma tecnologia para modelagem do software que a adotada pela TTBM visando reduzir o esforço e risco associado ao uso de diferentes tecnologias em um mesmo projeto.
	Uso de Modelos Intermediários	NÃO	Define se a TTBM adota ou não modelos intermediários entre o modelo comportamental/estrutural e os casos de teste para apoiar a geração automática de casos de teste. A existência de modelos intermediários pode introduzir um esforço adicional para a técnica, pois requer uma transformação do modelo original provido pelo processo de desenvolvimento para um novo modelo a ser construído pela equipe de teste.

Este corpo de conhecimento tem sido organizado e estruturado através de um banco de dados de uma ferramenta para gerência de referências bibliográficas, chamada JabRef (http://jabref.sourceforge.net), contendo todos os atributos de caracterização de TTBM (apresentados na Tabela 3.1) preenchidos com as informações extraídas dos artigos originais. Alguns atributos representam categorias com valores pré-definidos a fim de apoiar a classificação, agrupamento e consequente localização de TTBMs (ex: o nível de teste e tipo de técnica de teste), outros terão seus conteúdos livres possibilitando uma caracterização individual das técnicas (ex: restrições ou limitações).

Após a definição inicial da estrutura do corpo de conhecimento, foi identificada uma funcionalidade adicional que deveria compô-lo. Observou-se que cada atributo de caracterização que compõe o corpo de conhecimento pode ter um nível de relevância diferenciado no momento da seleção de TTBMs para um projeto de software. Assim, pesos devem ser associados a cada atributo quando estiver sendo realizada a tarefa de seleção de TTBMs.

No entanto, a definição dos pesos não pode ser feita aleatoriamente ou sem qualquer critério. Com isso, um *survey* foi planejado e executado visando à identificação do nível de relevância de cada atributo de caracterização de TTBMs no momento da seleção de técnicas para um projeto de software. O planejamento, projeto e alguns resultados deste *survey* foram publicados em (DIAS-NETO e TRAVASSOS, 2008b). As próximas subseções descrevem com mais detalhes o planejamento e os resultados obtidos ao longo deste estudo.

3.3.1 Definição e Planejamento do Survey

O objetivo deste estudo é observar quais das informações presentes em um conjunto de atributos que descrevem TTBMs, extraídos da literatura técnica (DIAS-NETO *et al*. 2007a; UTTING *et al*., 2006; PRETSCHNER *et al*., 2005; DALAL *et al*., 1999), podem ser pertinentes para caracterizar uma TTBM e quais dessas informações são relevantes para apoiar no momento da seleção de TTBMs para ser aplicada em um projeto de software sob a perspectiva de pesquisadores da área.

3.3.1.1 Definição do Objetivo do Estudo

Analisar *um conjunto de atributos que descrevem TTBMs extraídos da literatura técnica, principalmente dos trabalhos de (DALAL et al., 1999; DIAS-NETO et al., 2007a; PRETSCHNER et al., 2005; UTTING et al., 2006), e apresentados na Tabela 3.1.*

Com o propósito de *caracterizá-los*

Com respeito a *sua importância para caracterizar uma TTBM e sua correspondente relevância no contexto da seleção de TTBM para um projeto de software*

No ponto de visto de *pesquisadores trabalhando com pesquisa e desenvolvimento de TBM*

No contexto de *projetos de software que adotem TTBM*

Objeto de Estudo: conjunto inicial de atributos identificados em (DIAS-NETO *et al*, 2007) para apoiar a caracterização e comparação de TTBM.

3.3.1.2 Questões de Pesquisa

As questões de pesquisa associadas ao estudo estão relacionadas ao entendimento da importância e relevância dos atributos de caracterização de TTBMs identificados. Elas estão descritas a seguir:

Q1: Os atributos extraídos da literatura técnica, apresentados na Tabela 3.1 deste trabalho, são importantes para caracterizar uma TTBM?

Métrica 1.1: # de atributos de TTBM avaliados como “importante” para caracterizar uma TTBM de acordo com a opinião dos participantes do *survey*.

Q2: Existe algum atributo adicional considerado importante para caracterizar uma TTBM que não está presente no conjunto inicial?

Métrica 2.1: # de atributos adicionais a serem incluídos no conjunto inicial de acordo com a opinião dos participantes do estudo.

Q3: Existe algum atributo apresentado no conjunto inicial que não é importante para caracterizar uma TTBM?

Métrica 3.1: # de atributos a serem removidos da lista inicial de acordo com a opinião dos participantes do estudo.

Q4: Qual a ordem de relevância de todos os atributos de TTBM que compõem o conjunto final quando apoiando a seleção de uma TTBM para ser aplicada em um projeto de software?

Métrica 4.1: lista de todos os atributos de TTBM ordenada pelo nível de relevância.

3.3.1.3 Definição das Hipóteses

Foram definidas duas hipóteses nulas para este estudo que estão relacionadas, respectivamente, à (H_0 1) análise da importância dos atributos de caracterização e (H_0 2) suas relevâncias para a seleção de TTBMs. Elas estão descritas a seguir, junto com suas hipóteses alternativas:

Hipótese Nula 1 (H_0 1): O conjunto inicial de atributos de caracterização de TTBM está completo, ou seja, todos os atributos presentes neste conjunto são importantes para a caracterização de uma TTBM, e nenhum atributo foi incluído ou removido do conjunto.

- Aa – conjunto inicial de atributos de caracterização de TTBM
- Ar – atributos classificados como "não importante" que precisam ser removidos do conjunto inicial
- Ai – atributos não presentes no conjunto inicial e classificados como "importante" que precisam ser incluídos no conjunto final

$$\mathbf{H_0\ 1}: |Ar| = |Ai| = 0$$

Hipótese Alternativa (H_1): Existem atributos incluídos no conjunto inicial que foram classificados como "não importante" para a caracterização de uma TTBM. Portanto, eles precisam ser removidos do conjunto inicial.

$$\mathbf{H_1}: |Ar| \neq 0$$

Hipótese Alternativa (H_2): Existem atributos não presentes no conjunto inicial classificados como "importante" para a caracterização de uma TTBM. Portanto, eles precisam ser incluídos ao conjunto inicial.

$$\mathbf{H_2:}\ |Ai| \neq 0$$

Hipótese Nula (H_0 2): Todos os atributos de TTBM possuem mesmo nível de relevância relacionado ao apoio à seleção de uma TTBM para um projeto de software.

- RLi – Nível de Relevância relacionado ao apoio à seleção de uma TTBM para um projeto de software para o atributo "i", onde "i" é um número de 1 até n (número total de atributos avaliados de TTBM)

$$\mathbf{H_0\ 2:}\ RL_1 = RL_2 = \ldots = RL_n$$

Hipótese Alternativa (H_1): Existe ao menos um atributo de TTBM com diferente nível de relevância relacionado ao apoio à seleção de uma TTBM para um projeto de software.

$$\mathbf{H_1:}\ RL_i \neq RL_j$$

(onde "i" e "j" são números entre 1 e n, e "$i \neq j$")

3.3.1.4 Seleção de Contexto e de Participantes

A população do estudo adota foi o conjunto de autores de artigos científicos descrevendo TTBMs publicados na literatura e identificadas através da revisão sistemática descrita na Seção 3.2. Pelo fato de a revisão sistemática ter sido realizada buscando artigos publicados em inglês e por um longo período, acredita-se que o conjunto de autores de tais artigos representou de forma adequada a população de pesquisadores na área de Teste Baseado em Modelos. Ao total, 200 pesquisadores foram convidados para participar do estudo.

Os participantes do estudo foram contatados por e-mail. Neste e-mail de contato eles receberam um *login* e senha para acessar o questionário, evitando que participantes não planejados participassem do estudo, e consequentemente a introdução de viés nos resultados, por não fazerem parte da população deste estudo.

O estudo será realizado off-line, ou seja, o participante terá acesso ao website a qualquer momento para preencher o questionário sem qualquer monitoramento. Além disso, cada participante terá tempo ilimitado para responder ao questionário.

3.3.1.5 Variáveis

Variáveis Independentes:

- O conjunto inicial de atributos de caracterização de TTBMs.

Variáveis Dependentes:

- O conjunto final de atributos de caracterização de TTBMs.
- O nível de relevância para cada atributo de caracterização de TTBMs incluído no conjunto final com respeito à sua relevância para apoiar a seleção de uma TTBM para um projeto de software.

3.3.1.6 Definição da Instrumentação

Como instrumentação do estudo, um questionário on-line foi desenvolvido no idioma Inglês, permitindo que os participantes o acessassem através da Internet. O preenchimento do questionário é seguido em cinco passos.

1. **Tela de Login (Figura 3.1):** tela onde o participante faz sua autenticação.

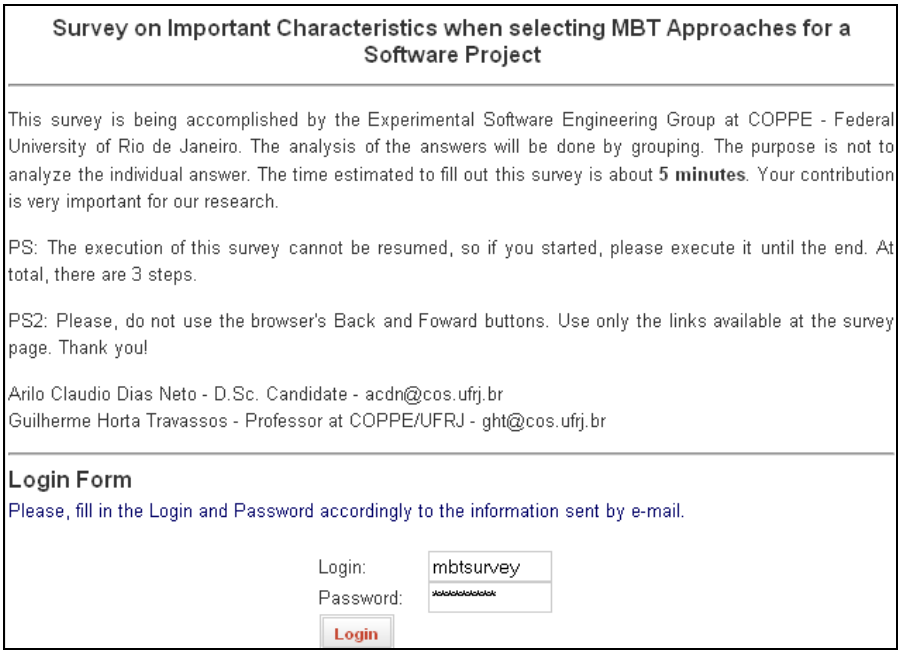


Figura 3.1. Tela de Login

2. **Caracterização do conhecimento e habilidades dos participantes (Figura 3.2).** Neste passo os participantes são questionados sobre seus dados pessoais (nome, e-mail, afiliação e país), nível de formação acadêmica, número de artigos publicados sobre TBM e nível de experiência em TBM na indústria;

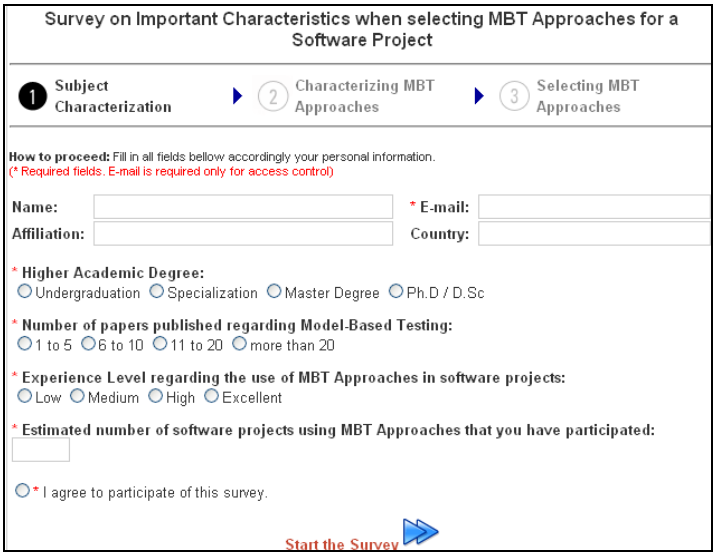


Figura 3.2. Tela de Caracterização e Autorização do Participante

3. **Identificação dos atributos importantes ou não importantes para caracterizar uma TTBM (Figura 3.3).** Para cada atributo, o participante deve preencher se ele é ou não importante para caracterizar uma TTBM, ou seja, se ele deve compor o corpo de conhecimento sobre TTBMs. Além disso, o participante pode inserir até 5 atributos adicionais que ele considere importantes e que não estão incluídos no conjunto inicial;

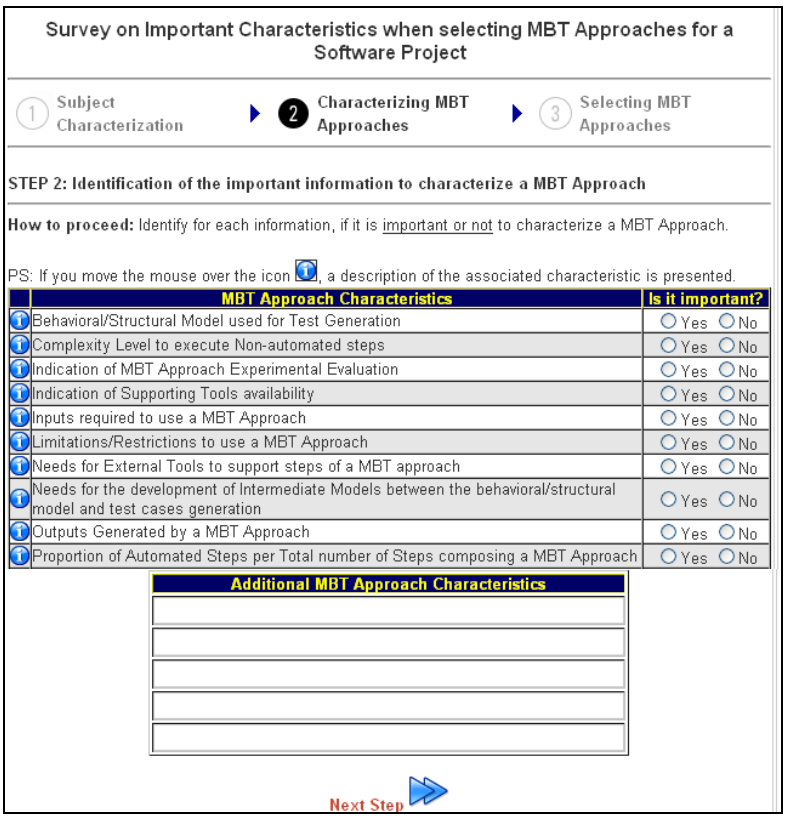


Figura 3.3. Tela de Identificação dos atributos importantes para caracterizar TTBM

4. **Definição do nível de relevância de cada atributo de caracterização de TTBM para a seleção de TTBMs para um projeto de software (Figura 3.4).** Neste passo, apenas os atributos indicados pelo participante como importantes no PASSO 2 são avaliados. Seis níveis de relevância foram definidos:

   (0) Sem Relevância (): é o nível de relevância mais baixo. Ele significa que o atributo não teria qualquer influência na seleção de uma TTBM para um projeto de software. A seleção de uma TTBM não seria afetada se o atributo em questão estiver ausente desta tarefa.

   (1) Muito Baixa Relevância (): indica que o atributo não afetaria significantemente no momento da seleção de uma TTBM para todos os projetos de software. Uma escolha incorreta não representaria um risco para projetos de software.

   (2) Baixa Relevância (): indica que a seleção de uma TTBM para um projeto de software seria mais precisa considerando também este atributo. No entanto, a tarefa de selecionar uma TTBM não sofreria seriamente com a ausência deste atributo. Em outras palavras, em alguns cenários particulares ela seria mais relevante, mas em geral a seleção não é afetada com a ausência deste atributo.

   (3) Média Relevância (): indica que o atributo possui efeito considerável no momento da seleção de uma TTBM para um projeto de software. Esta tarefa em certos projetos seria afetada com a ausência deste atributo. No entanto, uma incompatibilidade entre o atributo e a característica do projeto de software não tornaria inviável o uso da TTBM.

   (4) Alta Relevância (): indica que o atributo deve ser considerado no momento da seleção de uma TTBM para quase todos os projetos de software. Ele poderia ser desnecessário somente em um número restrito de cenários particulares onde ela não afetaria a seleção de uma TTBM.

   (5) Muita Alta Relevância (): indica que o atributo é absolutamente necessário de ser considerado no momento da seleção de uma TTBM para um projeto de software. Ausência (não uso) deste atributo no momento da escolha de uma TTBM pode inviabilizar as atividades de teste em um projeto software.

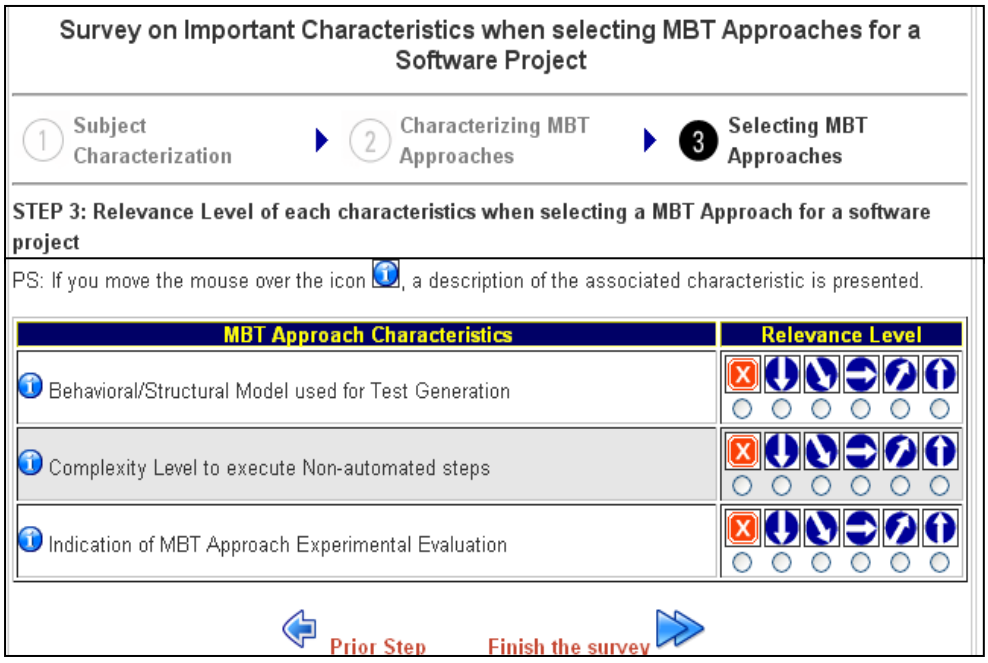


Figura 3.4. Tela de Definição do Nível de Relevância para a Seleção de TTBM

5. **Tela de Agradecimento (após a conclusão do estudo – Figura 3.5):** tela onde os pesquisadores agradecem aos participantes do estudo por sua colaboração.

Figura 3.5. Tela de Agradecimento

3.3.2 Execução do Survey

O questionário do estudo ficou ativo no período de 10/12/2007 a 10/02/2008 no endereço http://lens-ese.cos.ufrj.br. Os participantes do estudo foram contatados por e-mail. Neste e-mail de contato eles receberam um *login* e senha para acessar o questionário, evitando que participantes não planejados participassem do estudo, e consequentemente a introdução de viés nos resultados, por não fazerem parte da população deste estudo.

Dos 200 pesquisadores convidados para participar do estudo, 34 pesquisadores responderam ao *survey*. É preciso ressaltar que alguns fatores que não podem ser

confirmados ou mensurados podem ter afetado, possivelmente de forma negativa, o nível de confiança obtido com a quantidade de respostas recebidas. Entre eles:

- Os participantes foram convidados via e-mail, um mecanismo de comunicação assíncrono no qual não temos uma confirmação ou resposta se a pessoa recebeu ou não tal mensagem.
- Diversos e-mails enviados para os pesquisadores retornaram com mensagem de erro indicando que o endereço de e-mail estaria errado. Para vários deles foi realizada uma busca na Internet para se obter o endereço de e-mail atual, mas para 56 pesquisadores isso não foi possível.
- Alguns e-mails foram obtidos em artigos publicados há bastante tempo, o que não nos garante que eles ainda estejam sendo utilizados pelos pesquisadores que desejávamos contatar mesmo sem que eles tenham apresentado erros de endereço inválido, pois eles ainda podem existir, mas não estariam sendo utilizados.

No total, o envio de e-mail para 144 pesquisadores não resultou em falha, o que nos leva a crer que essa seria a nossa população de fato. Como obtivemos 34 respostas, isso nos dá uma confiança em torno de 85% dos dados coletados usando a fórmula de cálculo do nível de confiança de uma amostra descrita na Figura 3.6. No entanto, esse nível de confiança poderia ainda ser maior caso pudéssemos comprovar quantos pesquisadores efetivamente receberam o e-mail com o convite para participar do *survey*, pois assim teríamos nossa população real.

Onde:

- N = Tamanho da População
- E_0 = Nível de Confiança (em %)
- n = Tamanho da Amostra

$$n = \frac{N.\frac{1}{E_0^2}}{N + \frac{1}{E_0^2}} \rightarrow 34 = \frac{144.\frac{1}{E_0^2}}{144 + \frac{1}{E_0^2}} \rightarrow 85\%$$

Figura 3.6. Cálculo do Nível de Confiança de uma Amostra (HAMBURG, 1980)

Os participantes do estudo não serão analisados individualmente e não terão seus dados apresentados por questões de privacidade. Apenas uma análise geral será conduzida. Para a análise dos dados, cada participante deve possuir um peso diferenciado de acordo com seu nível de conhecimento e habilidade. Pesquisadores com níveis maiores de experiência ou habilidade devem possuir no estudo um peso maior. Após a definição dos pesos, as respostas de todos os participantes devem ser analisadas para cada atributo de caracterização de TTBMs avaliado, e ao final a ordem de relevância dos atributos será obtida.

A Tabela 3.2 apresenta os dados de caracterização dos participantes do estudo, incluindo o peso já calculado para cada participante.

Tabela 3.2. Caracterização dos Participantes do survey

ID	Tempo gasto (min)	Nível de Formação	Nº de artigos	Nível de experiência	Nº projetos usando TBM	Peso
1	07:48 min	Mestrado	1 a 5	2	2	5,5
2	07:33 min	Doutorado	1 a 5	1	1	5,25
3	06:03 min	Doutorado	6 a 10	2	3	7,75
4	03:12 min	Mestrado	1 a 5	2	4	6
5	08:04 min	Doutorado	mais que 20	3	10	12,5
6	08:43 min	Graduado	1 a 5	3	4	5
7	06:06 min	Doutorado	6 a 10	2	2	7,5
8	05:09 min	Doutorado	11 a 20	2	2	8,5
9	03:41 min	Doutorado	1 a 5	3	20	12
10	05:01 min	Doutorado	6 a 10	3	10	10,5
11	07:07 min	Doutorado	11 a 20	3	10	11,5
12	05:51 min	Mestrado	11 a 20	3	5	9,25
13	10:33 min	Doutorado	mais que 20	4	10	13,5
14	06:23 min	Doutorado	6 a 10	2	0	7
15	14:09 min	Doutorado	11 a 20	4	6	11,5
16	02:57 min	Doutorado	mais que 20	3	4	11
17	06:18 min	Doutorado	1 a 5	2	4	7
18	12:30 min	Mestrado	1 a 5	2	2	5,5
19	22:04 min	Mestrado	1 a 5	2	3	5,75
20	03:33 min	Doutorado	1 a 5	2	2	6,5
21	06:55 min	Mestrado	1 a 5	4	6	8,5
22	05:11 min	Mestrado	1 a 5	4	10	9,5
23	31:55 min	Doutorado	6 a 10	3	4	9
24	05:27 min	Doutorado	6 a 10	3	5	9,25
25	09:26 min	Doutorado	11 a 20	3	3	9,75
26	03:49 min	Doutorado	11 a 20	3	3	9,75
27	40:40 min	Mestrado	6 a 10	1	0	5
28	01:27 min	Doutorado	mais que 20	4	15	14,75
29	15:30 min	Mestrado	1 a 5	2	5	6,25
30	04:58 min	Doutorado	6 a 10	3	2	8,5
31	06:31 min	Mestrado	1 a 5	1	0	4
32	03:42 min	Doutorado	1 a 5	2	1	6,25
33	06:12 min	Mestrado	11 a 20	3	2	8,5
34	08:30 min	Doutorado	mais que 20	4	15	14,75

3.3.3 Análise dos Resultados do Survey

Após o cálculo dos pesos dos participantes, passou-se à análise da importância dos atributos avaliados para caracterização de TTBMs. Apresentar a resposta de cada participante para cada atributo aumentaria significantemente o tamanho deste trabalho e poderia dificultar o seu acompanhamento. Sendo assim, serão apresentados os dados já tabulados com os resultados obtidos após a fase de análise. Os detalhes e passos intermediários seguidos para análise dos dados obtidos estão descritos na seção seguinte.

3.3.3.1 Procedimentos de Análise

- **Ponderando um Participante do Estudo**

Para diferenciar as respostas de cada participante, serão atribuídos pesos de acordo com quatro perspectivas: a formação acadêmica, o número de artigos sobre TBM publicados, o nível de experiência na utilização de TBM em projetos de software, o número total de projetos de software em que participou onde foram utilizadas TTBMs. A fórmula usada para definir os pesos dos participantes foi baseada na proposta de DIAS-NETO (2006) para caracterizar engenheiros de software em um *survey* sobre práticas de Verificação e Validação, e foi adaptada para este estudo. A fórmula é:

$$Weight(i) = f(i) + p(i) + e(i) + \frac{t(i)}{MedianTP}, \text{ onde:}$$

- $Weight(i)$ é o peso atribuído ao participante *i*;
- $f(i)$ é a formação acadêmica. As opções para este campo são:
  $f(i)$ = 0, se o participante possui apenas nível superior;
  $f(i)$ = 1, se o participante possui especialização em engenharia de software;
  $f(i)$ = 2, se o participante possui mestrado;
  $f(i)$ = 3, se o participante possui doutorado;
- $p(i)$ é o indicador do número de artigos sobre TBM publicados pelo participante. As opções pra este campo são:
  $p(i)$ = 0, se o número de artigos está entre 1 e 5 artigos;
  $p(i)$ = 1, se o número de artigos está entre 6 e 10 artigos;
  $p(i)$ = 2, se o número de artigos está entre 11 e 20 artigos;
  $p(i)$ = 3, se o número de artigos é maior que 20;
- $e(i)$ é o indicador do nível de experiência sobre a utilização de TTBM em projetos de software. As opções pra este campo são:
  $e(i)$ = 0, se o nível de experiência é *baixo*;
  $e(i)$ = 1, se o nível de experiência é *médio*;
  $e(i)$ = 2, se o nível de experiência é *alto*;
  $e(i)$ = 3, se o nível de experiência é *excelente*;
- $t(i)$ é o indicador do número total (estimado) de projetos de software que usaram TTBM que o participante do estudo esteve envolvido.

- $MedianTP$ é a mediana do número total de projetos de software que usaram TTBM considerando as respostas de todos os participantes.

- **Identificando a Importância dos Atributos de TTBM**

Para definir quais atributos são importantes para caracterizar uma TTBM, é necessário primeiramente somar a resposta de todos os participantes (multiplicadas pelos seus respectivos pesos).

$$importância(j) = \sum_{i=1}^{M} (resposta(i, j) * peso(i))$$, onde:

- $importância(j)$ é o valor total das respostas de todos os participantes (multiplicadas pelos seus pesos) a respeito da importância do atributo *j* para caracterizar uma TTBM.
- $resposta(i, j)$ é o indicador de importância (1) ou não importância (0) definido pelo participante *i* para o atributo *j*
- $peso(i)$ é o peso atribuído ao participante *i*;
- *M* é o total de participantes que responderam ao *survey*

A definição se um atributo é importante ou não para caracterizar uma TTBM deve ser baseada em um ponto de corte, ou seja, um patamar indicando se o atributo deve ser incluído (valores acima do patamar definido) ou não (valores abaixo do patamar definido) no conjunto final. O patamar definido é 50% do valor máximo que pode ser obtido para um atributo *j* na variável $importância(j)$ se todos os participantes responderem "SIM" a respeito da importância de tal atributo para caracterizar uma TTBM.

$$Limite = 0{,}5 * \sum_{i=1}^{M} peso(i)$$, onde:

- $peso(i)$ é o peso atribuído ao participante *i*;
- *M* é o total de participantes que responderam ao *survey*

Portanto, o critério é:

- Se $importância(j) < Limite$ → atributo *j* é classificada como "não importante" e deve ser removida do conjunto
- Se $importância(j) \geq Limite$ → atributo *j* é classificada como "importante" e deve ser mantida no conjunto

- **Identificando a Ordem de Relevância dos Atributos de TTBM**

Para definir o nível de relevância de um atributo de caracterização de TTBM classificado previamente como “importante”, é necessário primeiramente somar a resposta de todos os participantes (multiplicadas pelos seus respectivos pesos).

$$Nrelevância(j) = \sum_{i=1}^{N} (Escala(i, j) * peso(i))$$, onde:

- $Nrelevância(j)$ é o valor total das respostas de todos os participantes (multiplicados pelos seus pesos) para o atributo *j*
- $Escala(i, j)$ é a escala de nível de relevância (0-5) definida pelo participante *i* para o atributo *j*
- *N* é o número total de participantes que responderam ao *survey*

Após este passo, os atributos serão ordenados. Os atributos mais relevantes serão aquelas com maior valor na variável $Nrelevância(j)$.

Para os novos atributos sugeridos pelos participantes, o seu nível de relevância foi calculado multiplicando o nível de relevância obtido considerando apenas os participantes que o sugeriram pelo valor médio de nível de relevância obtido entre todos os atributos originais do estudo.

3.3.3.2 Análise da Importância dos Atributos de Caracterização

Aplicando a fórmula definida para avaliar a importância ou não de um atributo de caracterização apresentada na Seção anterior, foram obtidos os resultados apresentados na Tabela 3.3.

Tabela 3.3. Avaliação da importância dos atributos de caracterização de TTBMs

Atributos de caracterização de TTBMs	Importância	Nível de Importância
Modelo Comportamental/Estrutural	292,75	100,00%
Critério de Geração de Casos de Teste	287,25	98,12%
Critério de Cobertura dos Testes	277,5	94,79%
Entradas requeridas para usar uma TTBM	276,75	94,53%
Limitações/Restrições para usar uma TTBM	272,5	93,08%
Característica de Qualidade de Software que a TTBM está apta a avaliar	265,5	90,69%
Nível de Teste	261,25	89,24%
Tipo de Técnica de Teste (Funcional ou Estrutural)	259,75	88,73%
Domínio de Software	250,25	85,48%
Resultados gerados pela TTBM	249,75	85,31%
Ferramenta de Apoio	246,75	84,29%
Resultados Históricos	235,75	80,53%
Uso de Modelos Intermediários	220,25	75,23%
Tecnologia de Geração dos Testes	214,75	73,36%
Nível de Complexidade dos Passos Não-automatizados	209,75	71,65%
Avaliação Experimental	204,25	69,77%
Necessidade de Ferramentas Externas	198,5	67,81%
Proporção de Passos Automatizados	192	65,58%

O limite inferior para um atributo ser considerado importante é 50%. Esse critério foi adotado por ser o ponto médio na escala de nível de importância (que varia de 0% a 100%), seguindo a fórmula para cálculo do nível de importância adotada. Dessa forma, todos os atributos avaliados inicialmente são considerados importantes para caracterizar uma TTBM, conforme pode ser observado na Figura 3.7.

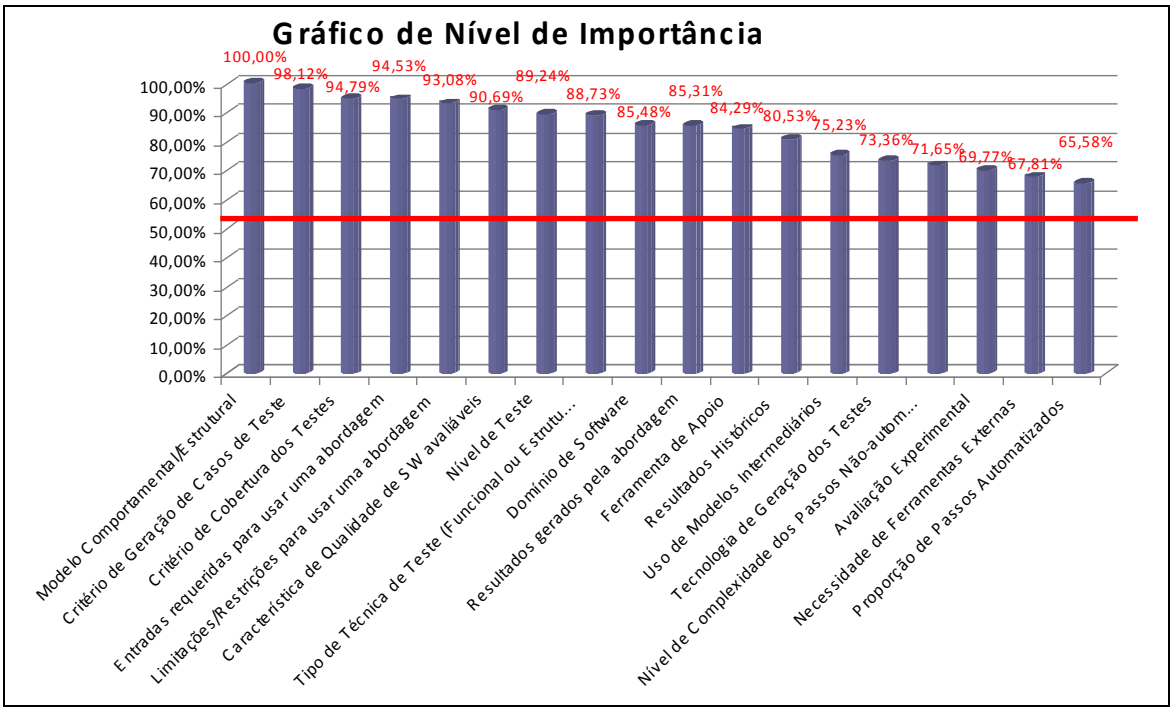


Figura 3.7. Gráfico de Nível de Importância

Além dos atributos presentes no conjunto inicial, outros 3 atributos foram adicionados ao conjunto a partir de indicações dos participantes do estudo. São eles:

- Habilidade/conhecimento necessário para usar a TTBM;
- Indicador da existência ou não de um mecanismo de rastreabilidade entre os requisitos do software, modelos e testes gerados;
- Indicador da existência ou não de um verificador de modelos compondo a TTBM.

Com isso, observa-se que a Hipótese Nula 1 (H_0 1) foi refutada, pois existiram atributos sugeridos pelos participantes que foram adicionados ao conjunto final inicial

3.3.3.3 Análise da Relevância dos Atributos de Caracterização

Após identificar que todos os atributos são considerados importantes pelos participantes do estudo, o passo seguinte é da definição dos seus graus de relevância para a seleção de TTBMs em projetos, ou seja, o peso de cada atributo para a seleção de TTBMs. Aplicando a fórmula para cálculo do nível de relevância apresentada na Seção 3.3.3.1, foram obtidos os resultados da Tabela 3.4.

Tabela 3.4. Avaliação da relevância dos atributos para seleção de TTBMs

Ordem	Atributo de caracterização de TTBMs	Nível de Relevância
1	Modelo Comportamental/Estrutural	83,48%
2	Critério de Cobertura dos Testes	74,79%
3	Critério de Geração de Casos de Teste	72,57%
4	Entradas requeridas para usar uma TTBM	72,54%
5	Limitações/Restrições para usar uma TTBM	69,96%
6	Tipo de Técnica de Teste (Funcional ou Estrutural)	69,89%
7	Resultados gerados pela TTBM	69,70%
8	Nível de Teste	69,70%
9	Característica de Qualidade de SW que a TTBM está apta a avaliar	68,30%
10	Ferramenta de Apoio	63,74%
11	Domínio de Software	55,85%
12	Uso de Modelos Intermediários	54,47%
13	Tecnologia de Geração dos Testes	52,54%
14	Necessidade de Ferramentas Externas	49,96%
15	Resultados Históricos	48,80%
16	Proporção de Passos Automatizados	47,48%
17	Avaliação Experimental	47,43%
18	Nível de Complexidade dos Passos Não-automatizados	46,58%
19	Existência de um mecanismo de rastreabilidade	34,39%
20	Habilidade/conhecimento necessário	33,00%
21	Existência de um verificador de modelos	18,71%

A Figura 3.8 apresenta todos os atributos ordenados de forma decrescente (da esquerda para a direita) de acordo com seu nível de relevância para apoiar a seleção de uma TTBM para um projeto de software.

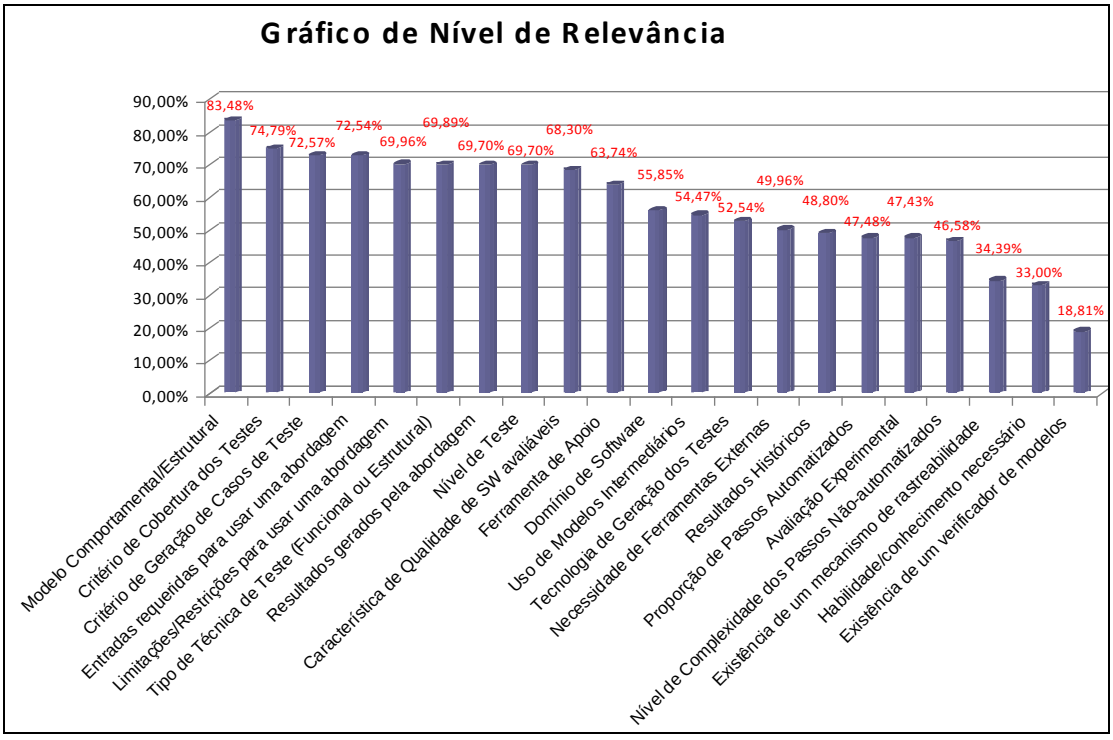


Figura 3.8. Gráfico de Nível de Relevância

Dessa forma, observa-se que a Hipótese Nula 2 (H_0 2) também foi refutada, uma vez que existiram atributos com diferentes níveis de relevância.

Observa-se que os resultados tendem a confirmar a percepção de importância de todos os atributos de caracterização de TTBMs avaliados pelos participantes do estudo[6] (mesmo aqueles não descritos com tanta intensidade nos artigos que descrevem tais técnicas, tais como: avaliação experimental ou grau de complexidade para usar uma TTBM). Além disso, os atributos de caracterização com maior grau de relevância no momento da seleção de TTBMs para um projeto de software são aqueles relacionados aos requisitos básicos para se utilizar uma técnica, tais como modelo estrutural/comportamental, critérios de geração/cobertura dos testes e restrições/limitações para se usar uma TTBM.

Com a análise dos resultados do *survey*, foi obtido um conjunto inicial de atributos usado para caracterizar uma TTBM e seus respectivos níveis de relevância no momento da seleção de TTBMs em um projeto de software (descritos na Tabela 3.4).

3.4 Considerações Finais do Capítulo

Com os resultados dos estudos descritos neste Capítulo, foi desenvolvida uma versão inicial do corpo de conhecimento sobre TTBMs. Esse elemento é fundamental na abordagem *Porantim* como mecanismo para viabilizar a seleção de TTBMs para projeto de software, pois funcionará como repositório de tais técnicas.

Este corpo de conhecimento necessita ser constamente atualizado para que não se tenha um impacto negativo na seleção das TTBMs em um projeto, por utilizar dados não atualizados sobre uma TTBM. Para sua atualização, estão previstas duas formas possíveis:

(1) No contexto de uma organização de software: edição dos dados sobre tais técnicas a partir do seu uso em projetos de software, através do uso da ferramenta JabRef e da biblioteca de TTBMs desenvolvida ao longo deste trabalho.

(2) No contexto de pesquisas na área de TBM: repetição dos dois estudos que serviram como base para a definição do corpo de conhecimento de TTBMs: a revisão sistemática (Seção 3.2) conduzida para definir o conteúdo do

[6] Lembrando-se que tais atributos de caracterização não são específicos de uma ou outra TTBM, mas sim aplicados a todas elas. Além disso, a lista inicial de atributos de caracterização avaliados não foi produzida pelos participantes do estudo.

corpo de conhecimento e o *survey* (S*eção* 3.3) conduzido para definir a sua estrutura. O pacote com o plano e diretrizes para condução dessas repetições fazem parte deste trabalho e foram descritos no Apêndice A e ao longo deste Capítulo.

A partir da elaboração de um corpo de conhecimento sobre TTBMs, este trabalho propõe uma abordagem, chamada *Porantim* (DIAS-NETO e TRAVASSOS, 2009a; DIAS-NETO e TRAVASSOS, 2009b), que visa apoiar a seleção de TTBMs para projetos de software. O capítulo seguinte apresenta a abordagem proposta e os elementos que a compõem, visando prover conhecimento que apóie a tomada de decisão a respeito de qual conjunto de TTBMs melhor se adéquam a um projeto de software.

CAPÍTULO 4 - *PORANTIM*: UMA ABORDAGEM PARA APOIAR A SELEÇÃO COMBINADA DE TÉCNICAS DE TESTE DE SOFTWARE BASEADA EM MODELOS

Neste capítulo será apresentada a abordagem Porantim, que provê apoio à seleção combinada de Técnicas de Teste Baseado em Modelos para projetos de software. Porantim é fundamentada em dois elementos básicos: um corpo de conhecimento sobre tais técnicas desenvolvido nesta pesquisa e descrito no Capítulo anterior, e um processo de apoio à sua seleção para projetos de software.

4.1 Visão Geral sobre a Abordagem *Porantim*

A seleção de tecnologias de software, independentemente de qual atividade do processo de desenvolvimento esteja sendo aplicada, pode depender de diversos fatores, dentre os quais podem ser citados fatores técnicos, tecnológicos, sociais, políticos e econômicos. A abordagem proposta que será descrita neste Capítulo utiliza como base alguns fatores técnicos (como tal tecnologia funciona na prática) e tecnológicos (em qual contexto tecnológico esta pode ser aplicada) com o objetivo de prover informações que possam apoiar na tomada de decisão a respeito da seleção de TTBMs em projetos de software. Ela se utilizará de características técnicas e tecnológicas do projeto de software, no qual se pretende aplicar Teste Baseado em Modelos, e das TTBMs disponíveis em um repositório, definido ao longo deste trabalho.

A abordagem proposta foi batizada de *Porantim* (DIAS-NETO e TRAVASSOS, 2009a; DIAS-NETO e TRAVASSOS, 2009b) e representa uma evolução da abordagem de apoio à seleção de técnicas de teste chamada de *Esquema de Caracterização*, proposta por VEGAS e BASILI (2005). Esta abordagem é fundamentada em dois elementos principais: (1) Corpo de Conhecimento sobre TTBMs e (2) Processo de Seleção de TTBMs (Figura 4.1).

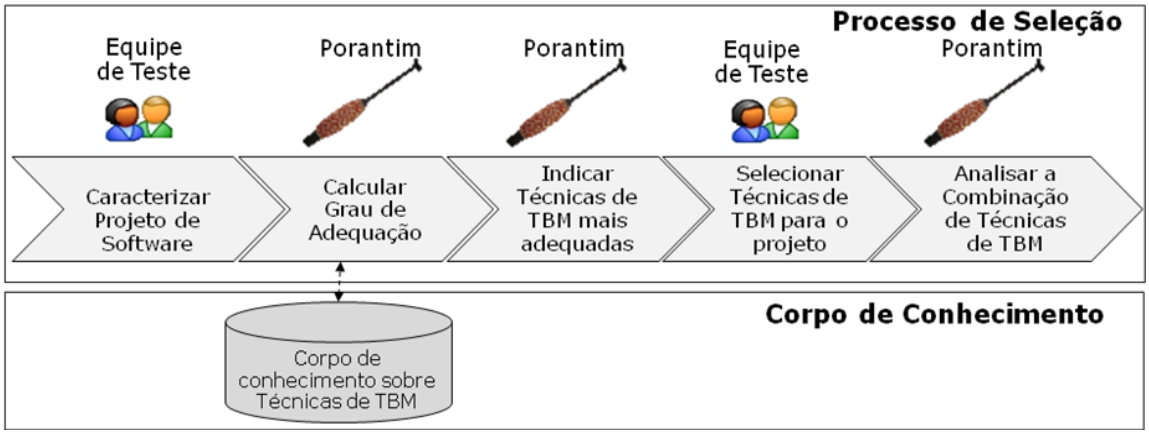


Figura 4.1. Esquema da Abordagem *Porantim* (DIAS-NETO e TRAVASSOS, 2009b)

1. **Corpo de Conhecimento sobre TTBM:** repositório de TTBMs, descrito no Capítulo 3. Cada técnica é caracterizada por um conjunto de atributos, sendo cada atributo ponderado em relação à sua importância para a seleção de técnicas. A definição dos atributos de caracterização de TTBMs e seus respectivos pesos para a seleção de tais técnicas foram apresentados no Capítulo anterior. Em relação à abordagem proposta por VEGAS e BASILI (2005), foram evoluídos os seguintes aspectos:
    a. **Evolução da caracterização de uma técnica de teste:** foram adicionados atributos de caracterização específicos para o domínio de Teste Baseado em Modelos, atributos adquiridos através de estudos experimentais. A premissa seguida é que a especialização dos atributos contribui de forma positiva na eficiência ou efetividade da seleção de um subconjunto de técnicas de teste, neste caso TTBMs, para um projeto de software.
    b. **Ponderação dos atributos de caracterização:** foram adicionados pesos aos atributos de caracterização. A premissa seguida é a de que cada atributo possui uma influência e significância diferenciada no processo de seleção de TTBMs.
2. **Processo de Seleção de TTBMs**[7]**:** conjunto de passos para direcionar e prover conhecimento para apoiar a seleção de TTBMs para projetos de software. As atividades que compõem tal processo, assim como as informações produzidas ao longo deste serão descritas na Seção 4.2. Os aspectos definidos no processo de seleção de TTBMs dizem respeito à:

[7] O Processo de Seleção de TTBMs se caracteriza como um processo de software, que segundo PRESSMAN (2005), consiste em uma seqüência coerente de práticas que objetiva apoiar em alguma tarefa relacionada ao desenvolvimento ou evolução de sistemas de *software*.

a. **Indicador de adequação entre TTBM e um projeto de software:** foi definida uma fórmula matemática para estimar o grau de adequação de uma TTBM em relação às características requeridas em um projeto de software. Para isso, usaram-se os novos atributos de caracterização adicionados e seus pesos.
b. **Indicadores de impacto da seleção combinada de TTBMs para um mesmo projeto de software:** foram definidas fórmulas matemáticas para estimar o impacto da seleção de mais de uma TTBM em algumas variáveis do processo de teste em um projeto de software.

A seção seguinte descreve o processo de seleção de TTBM, que consiste no segundo elemento que compõe a abordagem *Porantim* (o primeiro elemento – corpo de conhecimento – foi apresentado no Capítulo anterior).

4.2 Processo de Seleção de TTBMs

A existência de um corpo de conhecimento sobre TTBMs, descrito no Capítulo 3, é um passo importante para apoiar a seleção de TTBMs em projetos de software, porém possui função limitada, pois sozinho este provê muito conhecimento técnico sobre as TTBMs, porém nenhum direcionamento a respeito de como usá-lo para selecionar TTBMs mais coerentes ao contexto de um projeto de software. Conforme descrito na Seção 4.1, a abordagem *Porantim* é composta por um segundo elemento que visa apoiar na tarefa de seleção de TTBMs: um processo que é responsável por analisar as características do projeto de software a ser desenvolvido e das TTBMs disponíveis para uso, além de prover informações que apóiem na tomada de decisão sobre quais técnicas melhor se adéquam a um certo projeto de software. Tal processo é composto por cinco atividades (Figura 4.1) (DIAS-NETO e TRAVASSOS, 2009a). As próximas subseções irão descrever cada atividade que compõe este processo.

4.2.1 Caracterização do Projeto de Software

A primeira atividade que compõe o processo de apoio à seleção de TTBMs provido por *Porantim* consiste em conhecer as principais características do projeto de software no qual TTBMs serão aplicadas.

Esta caracterização deve ser realizada pela equipe de teste, a qual provê informações a respeito dos requisitos e restrições de teste para o projeto de software em questão. Tais características do projeto de software serão usadas como base para a seleção e indicação de TTBMs que mais se adéquam aos requisitos/necessidades do projeto de software.

A solução adotada para caracterização de um projeto de software neste trabalho consiste na definição de um conjunto de atributos de caracterização de projeto de software extraídos de diferentes fontes de pesquisa:

1) Ferramenta ADAPTPRO (BERGER, 2003), responsável pela definição de processos organizacionais e que compõe a Estação TABA[8]. Esta ferramenta possui uma caracterização de projetos de software a fim de sugerir modelos de ciclo de vida mais adequados, dadas as características de um projeto de software. Parte dessa caracterização foi adaptada para este trabalho;
2) Conjunto de atributos de caracterização de projetos e TTBMs obtidos a partir do *survey* que apoiou a definição da estrutura do corpo de conhecimento de TTBM descrito na Seção 3.3, e;
3) A partir do próprio desenvolvimento desta atividade do processo de seleção de TTBMs ao longo da pesquisa.

Os atributos de caracterização de projeto de software utilizados em *Porantim* estão apresentados na Tabela 4.1 junto com exemplos de possíveis valores que podem ser associados a eles. Tais atributos se dividem em 2 categorias: características do projeto de software a ser desenvolvido e requisitos de teste para o projeto de software.

Tabela 4.1. Atributos de Caracterização de Projeto de Software

Categoria	Atributos de Caracterização	Exemplo de Valores
Características do Projeto	1. Plataforma de execução do software	Ex: web, desktop, sw embarcado
	2. Paradigma de desenvolvimento adotado no projeto	Ex: Estruturado, Orientado a Objetos, Orientado a Aspectos
	3. Linguagem de programação adotada para construir o software	Ex: Java, C++, PHP, Phyton, Ruby
	4. Modelo comportamental/estrutural de software provido pelo projeto	Ex: Diagramas UML, Máquina de Estado Finito, Grafos
	5. Tecnologia usada para modelagem do software	Ex: UML, TSL, OCL
	6. Duração estimada para o projeto	Em meses
	7. Indicador de complexidade do problema	Alta, Média, Baixa
	8. Indicador de volatilidade dos requisitos	Alta, Média, Baixa
	9. Indicador de tamanho estimado da aplicação	Grande, Média, Pequena
	10. Habilidade provida pela equipe de teste alocada para o projeto	Ex: conhecimento em uma linguagem de programação ou de modelagem, conhecimento sobre um tipo de técnica de teste aplicável a uma plataforma de execução
Requisitos de Teste	11. Nível(is) de Teste desejado(s) para o projeto	Ex: unidade, integração, sistema, aceitação

[8] A Estação TABA consiste em uma infra-estrutura desenvolvida pela COPPE/UFRJ para auxiliar na definição, implementação e execução de Ambiente de Desenvolvimento de Software orientado a processos, organizações ou corporações (BERGER, 2003).

	12. Tipo de Técnica de Teste a ser aplicada	Ex: Funcional ou Estrutural
	13. Características de qualidade definidas nos requisitos e que devem ser avaliadas ao longo do projeto	Características de qualidade da norma ISO/IEC 9126 (ex: eficiência, funcionalidade, usabilidade)
	14. Tipos de Falhas que desejam ser reveladas	Ex: Tipo de Dados Incorreto, Falha de Banco de Dados, Falha de Navegação
	15. Apoio Ferramental requerido para testes	Não requer o uso de ferramenta, Apenas ferramentas gratuitas, Possibilidade de adquirir ferramenta
	16. Custo esperado para os testes	Alto, Médio, Baixo

A utilização deste conjunto de atributos de caracterização de projeto de software resultou na necessidade de evoluir a estrutura do corpo de conhecimento sobre TTBMs definido na Seção 3.3. Alguns atributos usados para caracterização de TTBMs precisaram ser divididos em outros atributos para que eles pudessem ficar compatíveis aos atributos usados para caracterização do projeto de software, conforme descrito na Tabela 4.2. Esses atributos se juntam aos demais atributos de caracterização e herdam o mesmo peso obtido pelo atributo que o originou e que já foram apresentados na Tabela 3.4.

Tabela 4.2. Evolução dos Atributos de Caracterização de TTBMs

Atributos da Versão Original	Atributos da Versão Atual	Peso
Domínio do Software	Plataforma de execução do software	55,85%
	Paradigma de desenvolvimento	55,85%
	Linguagem de programação adotada	55,85%
Ferramenta de Apoio	Custo associado à ferramenta de apoio	63,74%
	Nome da ferramenta de apoio	63,74%
	Plataforma em que a ferramenta de apoio opera	63,74%
Resultados Históricos	Tipos de Falhas que podem ser reveladas	48,80%
Limitações/Restrições para usar uma TTBM	Limitações/Restrições para usar uma TTBM	69,96%
	Habilidade requerida para ser operada	33,00%

A adição do atributo “Habilidade requerida para ser operada” tornou necessária a criação de uma nova categoria de atributos, pois este não se encaixava em nenhuma das categorias anteriores, descritas na Seção 3.3. Sendo assim, a categoria “Agente”, já adotada no equema de caracterização proposto por VEGAS e BASILI (2005), foi criada neste trabalho. Esta categoria considera as características dos usuários (testadores ou desenvolvedores) de uma TTBM. Esta informação pode ajudar futuramente a direcionar as pessoas mais adequadas a operarem uma técnica em um projeto de software.

4.2.2 Cálculo do Grau de Adequação de TTBMs

A partir da caracterização do projeto de software, a atividade seguinte do processo de seleção de TTBMs consiste na avaliação da adequabilidade das técnicas incluídas no corpo de conhecimento em relação ao projeto de software caracterizado.

Nesse momento, deve ser calculado um indicador chamado de *Grau de Adequação* entre o projeto de software caracterizado na atividade anterior e cada TTBM incluída no repositório (corpo de conhecimento). Este indicador é um valor numérico entre 0% e 100% e sugere quão próximas são as características de uma TTBM em relação às características do projeto de software, ou seja, quanto mais próximo de 100%, mais próxima seriam as características da TTBM em relação às características do projeto de software caracterizado.

Para o cálculo do *Grau de Adequação*, os atributos do projeto de software que podem ser confrontados com os atributos de caracterização de TTBMs (Tabela 3.4 + Tabela 4.2) possuem um peso específico, obtido através do *survey* citado na Seção 3.3 e publicado em (DIAS-NETO e TRAVASSOS, 2008b). De acordo com seus pesos, os atributos influenciam diferentemente no processo de seleção. A influência de cada atributo é calculada pela divisão do peso do atributo pela soma dos pesos de todos os atributos. Aqueles atributos que não podem ser confrontados com os atributos das TTBMs não serão usados para calcular o *Grau de Adequação*, e, portanto, terão peso e influência igual a 0. No entanto, eles compõem a abordagem para prover mais informações que podem ser consultadas para apoiar a tomada de decisão no momento da seleção das técnicas. A lista de atributos, seus pesos e influência estão apresentados na Tabela 4.3, e foram publicados em (DIAS-NETO e TRAVASSOS, 2009b).

Tabela 4.3. Atributos de Caracterização de Projeto de Software e seus pesos

Categoria	Atributos de Caracterização	Peso	Influência
Características do Projeto	1. Plataforma de execução do software	0,5585	8,50%
	2. Paradigma de desenvolvimento adotado no projeto	0,5585	8,50%
	3. Linguagem de programação adotada para construir o software	0,5585	8,50%
	4. Modelo comportamental/estrutural de software provido pelo projeto	0,8348	12,71%
	5. Tecnologia usada para modelagem do software	0,5254	8,00%
	6. Duração estimada para o projeto	Sem peso	--
	7. Indicador de complexidade do problema	Sem peso	--
	8. Indicador de volatilidade dos requisitos	Sem peso	--
	9. Indicador de tamanho estimado da aplicação	Sem peso	--
	10. Habilidade provida pela equipe de teste alocada para o projeto	0,3300	5,02%

Requisitos de Teste	11. Nível(is) de Teste desejado(s) para o projeto	0,6970	10,61%
	12. Tipo de Técnica de Teste a ser aplicada	0,6989	10,64%
	13. Características de qualidade definidas nos requisitos e que devem ser avaliadas ao longo do projeto	0,6830	10,40%
	14. Tipos de Falhas que desejam ser reveladas	0,4880	7,43%
	15. Apoio Ferramental requerido para testes	0,6374	9,70%
	16. Custo esperado para os testes		

A seguir será descrito como tal indicador é calculado no contexto desta pesquisa.

4.2.2.1 Representação Vetorial de Projeto de Software e TTBM

Para o cálculo do *Grau de Adequação* entre um projeto e uma TTBM está sendo utilizado o conceito matemático de distância euclidiana (BOLDRINI *et al*., 1980), que foi adotado por XAVIER *et al*. (2002) para apoiar a seleção de padrões arquiteturais em projetos de software. Segundo XAVIER *et al*. (2002), este mecanismo de busca explora a possibilidade de avaliação de distâncias conceituais a partir da comparação de elementos em um espaço vetorial multidimensional. A noção de distância conceitual é realizada, matematicamente, pela norma da diferença entre dois vetores *v1* e *v2*.

Neste trabalho, as características do projeto e de cada TTBM são transformadas em valores que posteriormente são representados através de vetores (*v1* – vetor das características do projeto de software) e (*v2* – vetor das características de cada TTBM). A distância entre eles indica o grau de adequação de uma TTBM para o projeto de software. Quanto mais próximos eles estiverem, mais adequada ao projeto é a TTBM (*grau de adequação* = |*1-distância*|**100*).

O cálculo da distância é realizado após a normalização dos vetores. Esta normalização é importante porque estamos interessados apenas na direção que cada vetor apresenta no espaço vetorial correspondente, minimizando a influência das dimensões dos vetores no cálculo da distância. Com a normalização, temos a equalização da importância das dimensões vetoriais e o enfoque fica apenas na direção que estes vetores podem assumir (KONTIO, 1995, apud XAVIER *et al*. 2002).

Alguns trabalhos mostram a possibilidade de utilização da abordagem vetorial para a avaliação de distâncias conceituais (MONETA *et al*., 1990; ASADA *et al*., 1992; XAVIER *et al*., 2002). Porém, foi preciso identificar de que forma poderíamos utilizar a abordagem vetorial para apoiar o processo de seleção de TTBMs para projetos de software. Ou seja, de que maneira esta abordagem poderia viabilizar a utilização do conhecimento obtido através dos estudos realizados ao longo deste trabalho, descritos na Seção 3.3 deste capítulo, para que a indicação de possíveis TTBMs possa atender da melhor forma possível ao projeto de software no qual desejamos aplicar TTBMs.

Percebemos que as próprias características do projeto e das TTBMs permitiriam uma representação vetorial, seguindo o relacionamento entre os atributos de caracterização apresentado na Tabela 4.4.

Tabela 4.4. Relacionamento entre Atributos de Caracterização de Projeto e TTBM

ID	Atributos de Caracterização do Projeto	Atributos de Caracterização de TTBM
1	Modelo comportamental/estrutural de software provido pelo projeto	Modelo Comportamental/Estrutural
2	Paradigma de desenvolvimento adotado no projeto	Paradigma de desenvolvimento
3	Linguagem de programação adotada para construir o software	Linguagem de programação adotada
4	Habilidade provida pela equipe de teste alocada para o projeto	Habilidade requerida para ser operada
5	Plataforma de execução do software	Plataforma de execução do software
6	Características de qualidade definidas nos requisitos e que devem ser avaliadas ao longo do projeto	Característica de Qualidade de Software que a TTBM está apta a avaliar
7	Apoio Ferramental requerido para testes	Nome da ferramenta de apoio
		Plataforma em que a ferramenta de apoio opera
	Custo desejado para os testes	Custo associado à ferramenta de apoio
8	Tecnologia usada para modelagem do software	Tecnologia de Geração dos Testes
9	Nível(is) de Teste desejado(s) para o projeto	Nível de Teste
10	Tipos de Falhas que desejam ser reveladas	Tipos de Falhas que podem ser reveladas
11	Tipo de Técnica de Teste a ser aplicado	Tipo de Técnica de Teste (Funcional / Estrutural) Resultados Históricos
12	Duração estimada para o projeto	---
13	Indicador de complexidade do problema	---
14	Indicador de freqüência de mudança dos requisitos	---
15	Indicador de tamanho estimado da aplicação	---

Sendo assim, cada atributo de caracterização do projeto de software e das TTBMs pode representar uma dimensão do espaço vetorial. O número total de dimensões é 11 (onze), que consiste no conjunto de atributos utilizado para caracterizar projetos de software e TTBMs que podem ser relacionados (conforme apresentado na Tabela 4.4):

$$V = R^{11} = (X_1 \mid X_2 \mid X_3 \mid X_4 \mid X_5 \mid X_6 \mid X_7 \mid X_8 \mid X_9 \mid X_{10} \mid X_{11})$$

onde V é o vetor que representa um projeto de software ou TTBM e X_i é a representação em número da característica do projeto ou TBTM referente ao atributo de índice i, sendo os atributos de caracterização de índice de 1 a 11 identificados na Tabela 4.4[9].

[9] Neste trabalho, apenas os atributos de caracterização de projeto de 1 a 11 são usados para definir a distância vetorial entre o projeto e TTBM. Os demais atributos de caracterização de projeto e de TTBM serão usados apenas para consulta e apoio à tomada de decisão por parte da equipe de teste.

Um ponto importante é que cada um dos 11 atributos de caracterização usado na representação vetorial de um projeto de software ou TTBM pode ser classificado como RESTRITIVO, ou seja, a sua combinação entre o projeto de software e a TTBM pode ser obrigatória, ocasionando um filtro apenas das TTBMs que atendem a este atributo de caracterização do projeto de software. Este filtro deve ser realizado antes do cálculo do grau de adequação da TTBM, realizado no passo seguinte.

4.2.2.2 Transformação de Características em Valores Numéricos e Cálculo da Distância Vetorial entre TTBM e Projeto de Software

Após definir a representação vetorial das características de projeto de software e das TTBM é preciso transformá-las em valores numéricos de forma a ser possível o cálculo vetorial das distâncias. Uma forma de tratamento desta quantificação de valores é a que atribui um valor a cada característica de acordo com a comparação entre a característica do projeto com a característica da TTBM. Nesta pesquisa, foi estabelecido que a faixa de valores utilizados em qualquer quantificação das características de um projeto ou TTBM seria definida no intervalo [0..1]. Além disso, cada atributo possui um peso de "importância" para a tarefa de seleção de TTBM para projetos de software identificado na Seção 3.3 e apresentado na Tabela 3.4.

Após a caracterização do projeto, cada característica do projeto é cruzada com a característica associada de uma TTBM, e a partir disso ocorre a transformação das características em valores numéricos entre 0 e 1, indicando a adequação ou não da característica da TTBM em relação à característica do projeto. A atribuição dos valores para características de um projeto de software ou TTBM foi realizada respeitando a seguinte regra:

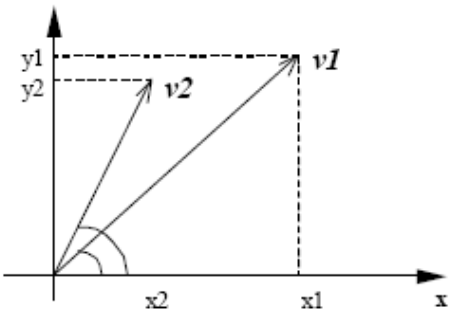
SE [atributo do projeto de software = atributo da TTBM] →
ENTÃO atributo da TTBM é transformado em 1 x influência do atributo (ver Tabela 4.3);
CASO CONTRÁRIO recebe o valor 0.

Apenas para o atributo de caracterização APOIO FERRAMENTAL essa regra é diferenciada, pois ele deve levar em consideração 2 atributos da TTBM:

SE [apoio ferramental do projeto de software = "Apoio Requerido"] e [TTBM possui apoio ferramental] →
ENTÃO SE [custo desejado para os testes no projeto = custo da TTBM]
ENTÃO recebe o valor 1 x influência do atributo (ver Tabela 4.3);
CASO CONTRÁRIO recebe o valor 0.
CASO CONTRÁRIO recebe o valor 0.

Uma vez definida a regra de transformação, o passo seguinte consiste na formalização da representação vetorial para projeto de software e TTBM e cálculo da distância entre os vetores.

O cálculo da distância vetorial entre o projeto de software e TTBM é feito através da fórmula apresentada na Figura 4.2.



v1: características de um projeto de software

v2: características de uma TTBM

Cálculo da Distância Vetorial:

$$d(v_1, v_2) = \sqrt{(x_1 - x_2)^2 + ... + (y_1 - y_2)^2}$$

Cálculo do *Grau de Adequação* de uma TTBM:

$$GrauAdequacao(v_1, v_2) = |1 - d(v_1, v_2)| * 100$$

Onde, “x” e “y” são atributos de caracterização de projeto/TTBM.

Figura 4.2. Fórmula para cálculo da distância entre vetores

4.2.2.3 Exemplo de Cálculo de Indicador de Grau de Adequação

Esta subseção apresenta um exemplo passo-a-passo descrevendo como é feita a caracterização de um projeto de software, como suas características são transformadas em valores numéricos e como é feito o cálculo dos indicadores de nível de adequação para duas TTBMs (não identificadas) extraídas do corpo de conhecimento.

- **Caracterização do Projeto e das TTBMs**

A Tabela 4.5 apresenta um exemplo de instanciação da caracterização de um projeto de software que está sendo desenvolvido no contexto do Grupo de Engenharia de Software Experimental da COPPE/UFRJ e que foi apresentado em dois relatos de experiência: (KALINOWSKI *et al*., 2007) e (DIAS-NETO *et al*., 2007c). Esse sistema está sendo desenvolvido em módulos e de forma incremental, onde cada módulo é tratado individualmente como um projeto. A Tabela 4.5 apresenta, ainda, a caracterização referente a duas TTBMs, identificadas apenas como *TTBM 1* e *TTBM 2*, extraídas do corpo de conhecimento sobre TTBMs descrito no Capítulo 3. Essas abordagens serão utilizadas para exemplificar o cálculo da distância entre projetos e TTBM apresentado anteriormente.

Tabela 4.5. Exemplo de Caracterização de Projeto de Software e TTBMs

Atributos de Caracterização	Projeto de Software	TTBM 1	TTBM 2
Modelo comportamental/ estrutural de software provido pelo projeto	Diagrama de Caso de Uso, Atividade, Estado e Classes	Diagrama de Estado e Colaboração	Diagrama de Casos de Uso, Atividade e Classes
Paradigma de desenvolvimento adotado no projeto	Orientado a objetos	Orientado a objetos	Orientado a objetos
Linguagem de programação adotada para construir o software	Java	Java	C++
Habilidade provida/requerida	Conhecimento em OO	Conhecimento em UML	Conhecimento em OO Conhecimento em UML
Plataforma de execução do software	Aplicação Web	Sistema Embarcado	Aplicação Web
Características de qualidade de software	Funcionalidade Eficiência	Funcionalidade Eficiência	Funcionalidade Eficiência Segurança
Apoio Ferramental requerido para testes	Ferramenta é requerida	Ferramenta A	Ferramenta B
Plataforma da ferramenta de apoio	Windows	Windows	Windows
Custo desejado para os testes	Não alocar recurso financeiro	Gratuita	Gratuita
Tecnologia usada para modelagem do software	UML	UML	UML
Nível(is) de Teste requerido/avaliado	Teste de Sistema e Teste de Unidade	Teste de Sistema	Teste de Unidade
Tipos de Falhas que desejam/podem ser reveladas	Erro Navegacional, Erro de Interface, Tipo de Dados Errado	Erro Navegacional, Tipo de Dados Errado	Erro Navegacional, Erro de Interface, Tipo de Dados Errado, Erro de Banco de Dados
Tipo de Técnica de Teste a ser aplicada/requerida	Funcional	Estrutural	Funcional

- **Obtenção dos Vetores**

A partir da caracterização do projeto e das TTBMs, os vetores resultantes, já com os pesos de cada atributo referente ao nível de relevância para a seleção de TTBMs para projeto, estão apresentados a seguir:

Representação Vetorial do Projeto e das técnicas $TTBM_1$ e $TTBM_2$

- $\Pr = (0{,}1271 \mid 0{,}0850 \mid 0{,}0850 \mid 0{,}0502 \mid 0{,}0850 \mid 0{,}1040 \mid 0{,}0970 \mid 0{,}0800 \mid 0{,}1061 \mid 0{,}0743 \mid 0{,}1064)$
- $TTBM_1 = (0{,}0635 \mid 0{,}0850 \mid 0{,}0850 \mid 0 \mid 0 \mid 0{,}1040 \mid 0{,}0970 \mid 0{,}0800 \mid 0{,}0530 \mid 0{,}0495 \mid 0)$
- $TTBM_2 = (0{,}1271 \mid 0{,}0850 \mid 0 \mid 0{,}0251 \mid 0{,}0850 \mid 0{,}1040 \mid 0{,}0970 \mid 0{,}0800 \mid 0{,}0530 \mid 0{,}0743 \mid 0{,}1064)$

Calculando-se a distância entre a representação vetorial do projeto e a representação vetorial de cada abordagem, temos:

Cálculo da distância e nível de adequação entre a $TTBM_1$ e o Projeto

- $d(\mathrm{Pr}, TTBM_1) = \sqrt{(0{,}0635)^2 + (0)^2 + (0)^2 + (0{,}0502)^2 + (0{,}0850)^2 + (0)^2 + (0)^2 + (0)^2 + (0{,}0530)^2 + (0{,}0248)^2 + (0{,}1064)^2}$
  - $d(\mathrm{Pr}, TTBM_1) = \sqrt{0{,}00403225 + 0{,}00252004 + 0{,}007225 + 0{,}002809 + 0{,}00061504 + 0{,}01132096} = 0{,}1688 \Rightarrow 83{,}12\%$ de adequação

Cálculo da distância e nível de adequação entre a $TTBM_2$ e o Projeto

- $d(\mathrm{Pr}, TTBM_2) = \sqrt{(0)^2 + (0)^2 + (0{,}0850)^2 + (0{,}0251)^2 + (0)^2 + (0)^2 + (0)^2 + (0)^2 + (0{,}0530)^2 + (0)^2 + (0)^2}$
  - $d(\mathrm{Pr}, TTBM_2) = \sqrt{0{,}007225 + 0{,}00063001 + 0{,}002809} = 0{,}1032 \Rightarrow 89{,}68\%$ de adequação

4.2.3 Indicação das TTBMs mais Adequadas e Seleção de TTBMs

Calculado o grau de adequação para as TTBMs incluídas no corpo de conhecimento, a atividade seguinte do processo de seleção de TTBMs consiste em ordenar de forma decrescente as técnicas por grau de adequação em relação ao projeto. Como existem muitas TTBMs disponíveis no corpo de conhecimento, o ideal é que apenas um subconjunto das TTBMs seja exibido (apenas as mais adequadas).

Para estas TTBMs mais adequadas, devem ser exibidos o seu indicador de grau de adequação e suas principais características, que levam a tal adequação. A Figura 4.3 exibe um exemplo apresentando indicadores dos graus de adequação das duas TTBMs utilizadas no exemplo da Seção 4.2.2.3.

	TTBM	Grau de Adequação (%)
☑	TTBM 1	83,12%
☑	TTBM 2	89,68%

Figura 4.3. Indicadores de grau de adequação de TTBM de acordo com o projeto

Exibidas as TTBMs mais adequadas, cabe à equipe de teste selecionar quais técnicas sugeridas por *Porantim* serão aplicadas no projeto de software em questão. A partir de tal seleção, *Porantim* irá prover na atividade seguinte uma análise sobre o impacto das TTBMs selecionadas em algumas variáveis do processo de testes, conforme descrito a seguir.

4.2.4 Análise do Impacto da Combinação de TTBMs no Processo de Testes

Como dito anteriormente, durante a atividade de seleção de TTBMs, mais que uma técnica pode ser selecionada, o que pode resultar em diferentes necessidades

para um projeto de software. Assim após a seleção das TTBMs que serão aplicadas em um projeto de software, no passo seguinte do processo de seleção de TTBMs, *Porantim* analisa o impacto da combinação de técnicas selecionadas em algumas variáveis do processo de testes.

Assim, para apoiar a análise dessas variáveis, foram definidas fórmulas matemáticas usando métricas obtidas durante a caracterização do projeto de software e das TTBMs. Cada variável é analisada considerando uma fórmula diferente para se obter os respectivos indicadores. Elas utilizam o conceito de *Coeficiente de Jaccard*[10] (JACCARD, 1901) e foram definidas considerando a experiência obtida pelo grupo ESE da COPPE/UFRJ na aplicação de técnicas de teste em projetos de software reais (DIAS-NETO *et al*., 2007c) e de pesquisas científicas conduzidas na área de TBM (DIAS-NETO *et al*., 2008). As variáveis analisadas são:

- **Indicador de Cobertura do Projeto de Software:** analisa quais requisitos de teste do projeto de software são atendidos pelas TTBMs selecionadas, tais como nível de teste, características de qualidade de software e tipos de falhas a serem reveladas.
- **Indicador de Esforço de Modelagem Salvo para a Criação dos Testes:** analisa quais modelos requeridos pelas TTBMs são disponibilizados a priori pelo processo de desenvolvimento de software e quais deles precisam ser construídos para possibilitar o uso das TTBMs selecionadas, caracterizando assim o esforço para construção dos modelos de teste.
- **Indicador de Grau de Preparação da Equipe de Teste:** analisa as habilidades providas pelos membros da equipe de teste selecionados para o projeto em relação às habilidades requeridas para uso das TTBMs selecionadas.

As próximas subseções irão descrever como cada indicador pode ser calculado.

4.2.4.1 Indicador de Cobertura do Projeto de Software

Este indicador visa a identificar qual a proporção das necessidades do projeto que estão sendo atendidas pelas TTBMs selecionadas, identificando inclusive quais técnicas fornecem apoio a algo não requerido pelo projeto (algo que poderia resultar em

[10] *Coeficiente de Jaccard* (1901) compara o número de elementos similares e o total de elementos em um conjunto (chamado coeficiente de *similaridade*).

um esforço extra). Ele considera três aspectos para avaliar tal cobertura, que são: nível de teste, características de qualidade de software e tipos de falhas a serem reveladas.

- **Análise de Nível de Teste:**
  - ✓ *Projeto de software requer um nível de teste provido pelas TTBMs selecionadas:* este é o cenário ideal e, portanto, não existe qualquer impacto negativo no projeto de software.
  - ✓ *Projeto de software requer um nível de teste, mas ele não é provido pelas TTBMs selecionadas*: este não é o cenário desejado (existe algo requerido, mas não provido) e, portanto, existe um impacto negativo no projeto de software.
  - ✓ *Projeto de software não requer um nível de teste, mas ele é provido pelas TTBMs selecionadas*: este não é o cenário desejado (será necessário investir um esforço adicional para realizar tal nível de teste não requerido) e, portanto, existe um impacto negativo no projeto de software.

Assim, a fórmula usada para calcular o indicador de nível de teste está apresentada na Figura 4.4, onde:

- ✓ *NivelT(proj)* é o conjunto de nível(is) de teste requeridos pelo projeto.
- ✓ *NivelT(teqs)* é o conjunto de níveis de teste atendidos pelas TTBMs selecionadas.
- ✓ *#[NivelT(proj) – NivelT(teqs)]* é o número de níveis de teste diferentes requeridos pelo projeto, mas não atendido pelas TTBMs selecionadas.
- ✓ *#[NivelT(teqs) – NivelT(proj)]* é o número de níveis de teste diferentes providos pelas TTBMs selecionadas, mas requeridos pelo projeto.
- ✓ *#[NivelT(proj) ∪ NivelT(teqs)]* é o número de diferentes níveis de teste requeridos pelo projeto OU providos pelas TTBMs selecionadas.

$$NivelT = 1 - \left(\frac{\#[NivelT(proj) - Nivel(teqs)] + \#[NivelT(teqs) - NivelT(proj)]}{\#[NivelT(proj) \cup NivelT(teqs)]}\right)$$

Figura 4.4. Fórmula para Calcular o Indicador de Nível de Teste

- **Análise das Características de Qualidade de Software:**
  - ✓ *Projeto de software requer uma característica de qualidade de software que pode ser avaliada pelas TTBMs selecionadas:* este é o cenário ideal e, portanto, não existe qualquer impacto negativo no projeto de software.

- ✓ *Projeto de software não requer uma característica de qualidade de software, mas ela consiste em uma característica de qualidade que pode ser avaliada pelas TTBMs selecionadas:* este é um aspecto adicional que não possui qualquer impacto negativo no projeto de software.
- ✓ *Projeto de software requer uma característica de qualidade de software, mas ela consiste em uma característica de qualidade que não pode ser avaliada pelas TTBMs selecionadas:* este não é o cenário desejado (existe algo requerido, mas não provido) e, portanto, existe um impacto negativo no projeto de software.

Assim, a fórmula usada para calcular o indicador de característica de qualidade de software está apresentada na Figura 4.5, onde:

- ✓ *CaracQSW(proj)* é o conjunto de características de qualidade de software requeridas pelo projeto de software.
- ✓ *CaracQSW (teqs)* é o conjunto de características de qualidade de software que podem ser avaliadas pelas TTBMs selecionadas.
- ✓ *#CaracQSW(proj)* é o número de características de qualidade de software requeridas pelo projeto de software.
- ✓ *#[CaracQSW(proj) ∩ CaracQSW(teqs)]* é o número de diferentes características de qualidade de software requeridas pelo projeto de software E que podem ser avaliadas pelas TTBMs selecionadas.

$$CaracQSW = \frac{\#[CaracQSW(proj) \cap CaracQSW(teqs)]}{\#CaracQSW(proj)}$$

Figura 4.5. Fórmula para Calcular o Indicador de Característica de Qualidade de Software

- **Análise dos Tipos de Falhas a serem reveladas:**
  - ✓ *Projeto de software requer a detecção de um tipo de falha provido pelas TTBMs selecionadas:* este é o cenário ideal e, portanto, não existe qualquer impacto negativo no projeto de software.
  - ✓ *Projeto de software não requer a detecção de um tipo de falha, mas ele é provido pelas TTBMs selecionadas:* este é um aspecto adicional que não possui qualquer impacto negativo no projeto de software.
  - ✓ *Projeto de software requer a detecção de um tipo de falha, mas ele não é provido uma característica de qualidade de software:* este não é o cenário desejado (existe algo requerido, mas não provido) e, portanto, existe um impacto negativo no projeto de software.

Assim, a fórmula usada para calcular o indicador de tipos de falhas está apresentada na Figura 4.6, onde:

- ✓ *Falhas(proj)* é o conjunto de tipos de falhas requeridas a serem avaliadas no projeto de software.
- ✓ *Falhas(teqs)* é o conjunto de tipos de falhas que podem ser revelados pelas TTBMs selecionadas.
- ✓ *#Falhas(proj)* é o número de tipos de falhas requeridas a serem avaliadas no projeto de software.
- ✓ *#[Falhas(proj) ∩ Falhas(teqs)]* é o número de tipos de falhas diferentes requeridas a serem avaliadas no projeto de software E que podem ser revelados pelas TTBMs selecionadas.

$$Falhas = \frac{\#[Falhas(proj) \cap Falhas(teqs)]}{\#Falhas(proj)}$$

Figura 4.6. Fórmula para Calcular o Indicador de Tipos de Falhas

Para obter o indicador de *Cobertura do Projeto de Software*, é calculada a média entre os três valores previamente obtidos de acordo com a fórmula apresentada na Figura 4.7.

$$CoberturaPSW = \frac{NivelT + CaracQSW + Falhas}{3}$$

Figura 4.7. Fórmula para Calcular o Indicador de Cobertura do Projeto de Software

O resultado obtido deve ser um valor entre 0 e 100%. Um exemplo de cálculo deste indicador está apresentado na Tabela 4.6.

Tabela 4.6. Exemplo de Indicador de Cobertura do Projeto de Software

Atributo	Projeto de Software	TTBM 1	TTBM 2	Cálculo
Nível de Teste	Teste de Sistema e Integração	Teste de Sistema e Unidade	Teste de Sistema e Integração	$1-(\frac{0+1}{3}) = 0{,}66$
Característica de Qualidade de Software	Funcionalidade, Segurança, Desempenho, Usabilidade	Funcionalidade, Desempenho, Usabilidade	Funcionalidade	$\frac{3}{4} = 0{,}75$
Tipo de Falhas	Tipo 1, Tipo 2, Tipo 3	Tipo 2, Tipo 4	Tipo 1	$\frac{2}{3} = 0{,}66$
Cobertura do Projeto de Software	$\frac{0{,}66 + 0.{,}75 + 0{,}66}{3} = 0{,}69 \rightarrow 69\%$			

4.2.4.2 Indicador de Esforço de Modelagem Salvo para a Criação dos Testes

Este indicador visa identificar uma proporção de quais modelos requeridos pelas TTBMs já estão sendo providos pelo projeto, não resultando em esforço

adicional, e quais ainda precisam ser construídos para que as TTBMs selecionadas possam ser utilizadas, o que resulta em esforço adicional. Cada modelo possuirá um peso diferenciado no cálculo deste indicador, de acordo com a sua combinação entre estar disponível no processo de desenvolvimento e ser requerido por 1 ou mais TTBMs. Cinco diferentes cenários são considerados na Figura 4.8. Foi usada uma escala ordinal (de 0 até 4) para representar diferentes pesos durante a análise de esforço de modelagem para os testes. A Figura 4.8 descreve um possível cenário de combinação entre os modelos providos pelo processo de desenvolvimento (retângulo da esquerda) e duas TTBMs (retângulo central e da direita). Nesta figura, cada ícone não possui qualquer significado maior e está sendo utilizado apenas para representar um tipo de modelo diferente. A sua presença em um retângulo indica se o modelo é provido pelo processo de desenvolvimento e/ou é requerido pelas TTBMs selecionadas para o projeto.

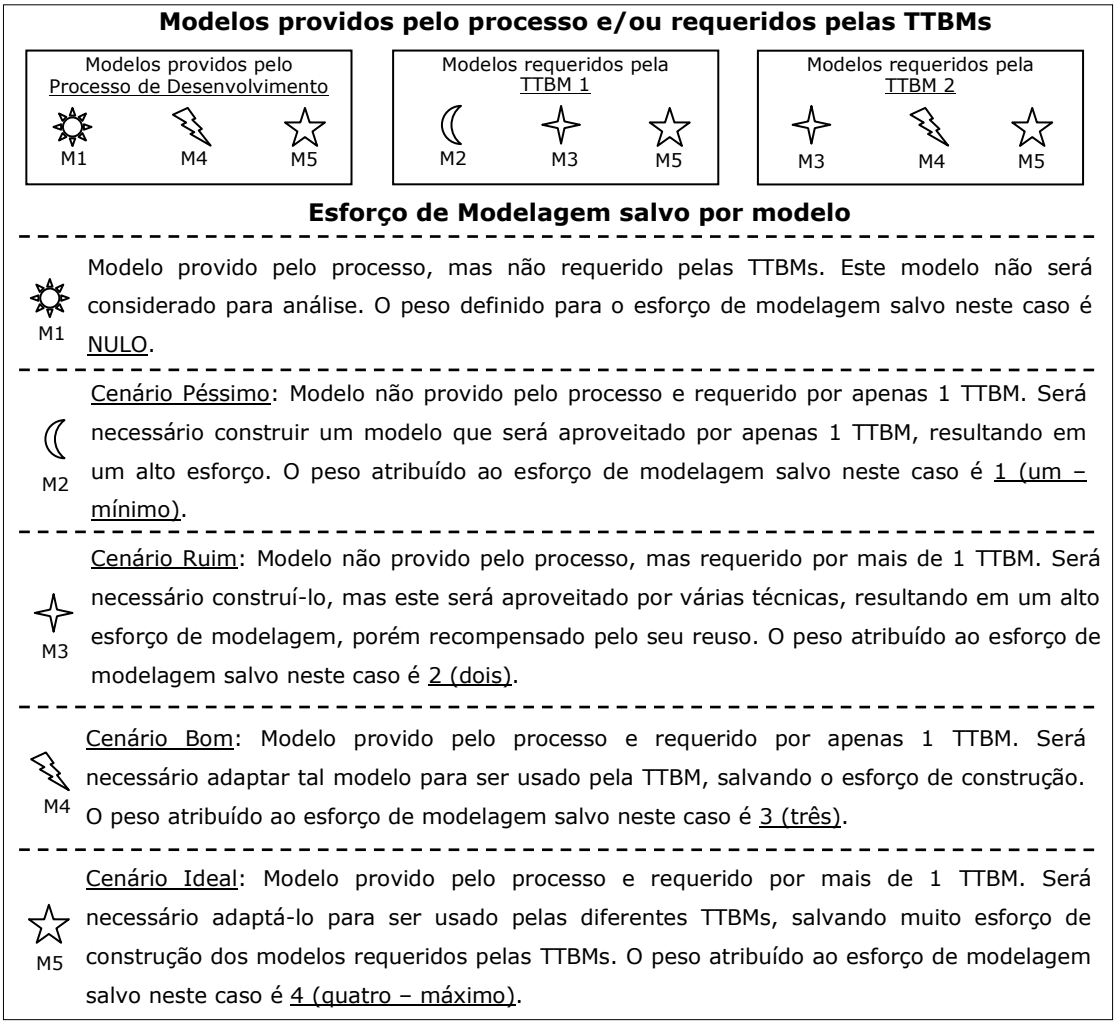


Figura 4.8. Pesos dos modelos para cálculo do esforço de modelagem salvo

Assim, também através do uso do *Coeficiente de Jaccard*, a fórmula usada para calcular o esforço de modelagem salvo para criação dos testes corresponde ao

total de modelos contados em cada categoria descrita acima que prevê um esforço de modelagem (M2 até M5[11]) multiplicado pelos seus respectivos pesos e dividido pelo número total de modelos. Esta fórmula está apresentada na Figura 4.9, onde *#MX* corresponde ao número de modelos usado por uma TTBM ou provido no projeto de software em uma das categorias descritas previamente[11] (M2, M3, M4 ou M5).

$$Esforço = \frac{(\#M2*1) + (\#M3*2) + (\#M4*3) + (\#M5*4)}{(\#M2+\#M3+\#M4+\#M5)*4}$$

Figura 4.9. Fórmula para Calcular o Indicador de Esforço de Modelagem Salvo

O valor obtido também é um número entre 0 e 100%. Considerando o cenário apresentado na Figura 4.8, o indicador de esforço de modelagem salvo para criação dos testes está calculado e visualizado na Figura 4.10.

$$Esforço = \frac{(1*1) + (1*2) + (1*3) + (1*4)}{4*4} = \frac{10}{16} = 0{,}625 \rightarrow 62{,}5\%$$

Figura 4.10. Exemplo de Indicador de Esforço de Modelagem Salvo

4.2.4.3 Indicador de Grau de Preparação da Equipe de Teste

Este identificador visa a analisar que habilidades necessárias para se usar as TTBMs selecionadas (este conjunto será chamado pela variável *TTBM*) são providas por uma determinada configuração de equipe de teste que foi definida durante o planejamento de um projeto de software (o conjunto de habilidades providas pela equipe de teste será chamado pela variável *Equipe*). Tal análise possibilita avaliar a adequação da equipe de teste selecionada para o projeto em relação às TTBMs selecionadas, e assim avaliar a necessidade de treinamento da equipe de teste. A fórmula a ser usada está descrita na Figura 4.11.

$$\mathrm{Re}cursosHumanos = \frac{|Equipe \cap TTBM|}{|TTBM|}$$

Figura 4.11. Fórmula para Calcular o Indicador de Recursos Humanos

O resultado obtido por este indicador também deve ser um valor entre 0 e 100%. Um exemplo de cálculo deste indicador está apresentado na Figura 4.12.

[11] Conforme descrito anteriormente, a categoria "M1" não introduz nenhum esforço, por isso não é considerada no cálculo do indicador de esforço de modelagem.

Habilidades providas pela equipe de teste e Habilidades requeridas pelas TTBMs

Recursos Humanos	Habilidades Providas
RH #1	Habilidade A, *Habilidade B*, *Habilidade C*, *Habilidade D*
RH #2	*Habilidade C*, *Habilidade D*, Habilidade E

Técnicas de TBM	Habilidades Requeridas
TTBM #1	*Habilidade* B, *Habilidade C*
TTBM #2	*Habilidade C*, *Habilidade D*, Habilidade F

Cálculo do Indicador de Recursos Humanos

Conjuntos	Habilidades
[Equipe ∩ TTBM]	Habilidade B, Habilidade C, Habilidade D
[TTBM]	Habilidade B, Habilidade C, Habilidade D, Habilidade F

$$\text{Re}cursosHumanos = \frac{3}{4} = 0.75 \rightarrow 75\%$$

Figura 4.12. Exemplo de Análise do Indicador de Recursos Humanos

4.2.4.4 Análise dos Indicadores

A partir do cálculo dos três indicadores, o responsável pela seleção das TTBMs pode analisar qual a melhor combinação para um dado projeto, tentando sempre maximizar os três indicadores obtidos. A Tabela 4.7 apresenta um cenário fictício de análise da combinação de 3 possíveis TTBMs para um projeto. Neste exemplo, podemos observar que a seleção individual de cada TTBM resulta em valores baixos para os indicadores calculados. Podemos também perceber que a combinação da TTBM 1 com a TTBM3 não resultou em nenhuma melhoria, provavelmente pelo fato de elas possuírem características muitos similares, de forma que estaríamos sendo redundante ao aplicar as duas técnicas simultaneamente, ou seja, elas devem cobrir os mesmos aspectos de um software. Podemos ainda perceber que a combinação das 3 TTBMs não é a melhor opção, pois se por um lado aumenta a cobertura do projeto de software, reduz o esforço de modelagem salvo (provavelmente porque as TTBMs utilizam modelos diferentes, alguns deles não providos pelo projeto) e o grau de preparação da equipe de teste (pois juntas podem requerer uma habilidade não provida pela equipe de teste). Sendo assim, a melhor combinação para este cenário seria adotar as TTBMs #2 e #3.

Tabela 4.7. Exemplo Fictício 01 de Seleção Combinada de TTBMs

Combinação de TTBMs	Indicadores Obtidos		
	Cobertura do Projeto de Software	Esforço de Modelagem Salvo	Grau de Preparação da Equipe de Teste
TTBM 1	50%	20%	70%
TTBM 2	85%	70%	85%
TTBM 3	70%	65%	50%
TTBM 1 + TTBM 2	92%	80%	85%
TTBM 1 + TTBM 3	70%	65%	50%
TTBM 2 + TTBM 3	95%	93%	90%
TTBM 1 + TTBM 2 + TTBM 3	95%	80%	85%

No entanto, podem ocorrer situações em que não haja uma combinação ideal que indique um maior valor para os três indicadores. Nesse caso, cabe ao responsável pela tarefa de seleção das TTBMs escolher quais variáveis são mais importantes de serem consideradas para a tomada de decisão no projeto em questão.

Por exemplo, olhando para o cenário da Tabela 4.8, se há a possibilidade de treinar a equipe de teste para um determinado projeto, o responsável pela seleção pode optar por uma combinação de técnicas que maximize a cobertura do projeto de software e o esforço de modelagem salvo, sem considerar o terceiro indicador (combinação TTBMs 1 e 3). Ou pode acontecer de em uma situação cada combinação maximizar um indicador diferente, o que dificulta ainda mais a seleção. Por isso, cabe ao responsável pela tarefa analisar quais as prioridades para o projeto em questão, para então decidir por quais TTBMs irá selecionar para um projeto de software.

Tabela 4.8. Exemplo Fictício 02 de Seleção Combinada de TTBMs

Combinação de TTBMs	Indicadores Obtidos		
	Cobertura do Projeto de Software	Esforço de Modelagem Salvo	Grau de Preparação da Equipe de Teste
TTBM 1	70%	60%	70%
TTBM 2	45%	70%	85%
TTBM 3	80%	45%	50%
TTBM 1 + TTBM 2	76%	80%	85%
TTBM 1 + TTBM 3	92%	76%	50%
TTBM 2 + TTBM 3	84%	72%	90%
TTBM 1 + TTBM 2 + TTBM 3	92%	70%	85%

4.3 Considerações Finais do Capítulo

Neste Capítulo, foram definidas as atividades que compõem o processo de apoio à seleção de TTBMs proposto pela abordagem *Porantim*. A existência deste processo de apoio à seleção de TTBMs representa uma das principais evoluções providas por *Porantim* em relação à abordagem proposta por VEGAS e BASILI (2005), que conforme dito anteriormente foi a base utilizada para o desenvolvimento da abordagem *Porantim* proposta neste trabalho. Entre as principais características presentes neste processo e que tornam *Porantim* uma abordagem original no contexto de seleção de tecnologias de software, mas precisamente TTBMs, podem ser citadas:

- Os atributos de caracterização das TTBMs são específicos para o domínio em questão e possuem pesos diferenciados entre si, obtidos a partir dos resultados de um estudo experimental (DIAS-NETO e TRAVASSOS, 2008b), o que possibilita prover indicação mais especializada a respeito de quais TTBMs seriam melhor adequadas a um projeto de software.

- É calculado um indicador numérico de adequação entre as TTBMs e o projeto de software a partir do cruzamento dos atributos de caracterização das TTBMs e do projeto de software, considerando seus respectivos pesos, utilizando o conceito matemático de distância Euclidiana entre vetores. Tal indicador possibilita ordenar as TTBMs a partir de sua adequabilidade a um projeto de software.
- E principalmente, após a seleção das TTBMs pela equipe de teste, é provida uma análise a indicando o impacto da combinação das TTBMs selecionadas em variáveis do processo de teste, ou seja, *Porantim* provê apoio para a seleção simultânea de mais de uma TTBM para um mesmo projeto de software.

No próximo Capítulo será descrita a avaliação experimental que teve como objetivo avaliar *Porantim* em relação a outra técnica de apoio à seleção de técnicas de teste proposta em VEGAS e BASILI (2005).

CAPÍTULO 5 - AVALIAÇÃO EXPERIMENTAL DA ABORDAGEM *PORANTIM*

Neste capítulo serão apresentados estudos experimentais com o propósito de avaliar a abordagem proposta de apoio à seleção de Técnicas de Teste de Software Baseada em Modelos, a fim de observar se as questões de pesquisa definidas foram respondidas.

5.1 Objetivo do Estudo Experimental

Seguindo a metodologia científica adotada para o desenvolvimento da abordagem *Porantim*, descrita na Seção 1.5, após a fase de concepção da tecnologia de software que está sendo desenvolvida, a fase seguinte consiste em sua avaliação a partir da condução de estudos experimentais, a fim de avaliar sua viabilidade e efetividade em relação ao que ela se propõe. No caso desta pesquisa, estudos experimentais foram conduzidos a fim de se avaliar a viabilidade e efetividade da abordagem *Porantim* em relação à seleção de TTBMs em projetos de software, considerando as questões de pesquisa definidas na Seção 1.3.

Estes estudos foram conduzidos como uma repetição externa (re-execução de um estudo conduzido por outro grupo de pesquisa [MENDONÇA *et al*., 2008]) do estudo experimental conduzido por VEGAS e BASILI (2005), cujo objetivo foi a avaliação de uma abordagem de seleção de técnicas de teste de software, em geral.

Sendo assim, o objetivo deste estudo é avaliar aspectos de interesse para engenheiros de software usando a abordagem *Porantim*, como completude, efetividade, eficiência, usabilidade e satisfação do usuário, quando comparada a outra abordagem que apóia a seleção de TTBMs. Para este estudo são avaliadas 2 abordagens de seleção de TTBMs: *Porantim*, proposta nesta pesquisa, e o Esquema de Caracterização, proposto por VEGAS e BASILI (2005). A abordagem Esquema de Caracterização foi usada para comparação neste estudo por ter sido a abordagem base que proveu os fundamentos para a definição da abordagem *Porantim*. Dessa forma, um dos propósitos do estudo é observar se as funcionalidades adicionais providas por *Porantim* (processo de seleção e características específicas de TTBMs) possuem influência no processo de seleção quando comparado com o uso de uma abordagem de seleção mais "genérica". Além disso, foi possível re-aproveitar o planejamento e instrumentos do estudo que avaliou esta abordagem proposta por

VEGAS e BASILI (2005) em relação ao uso de livros e artigos durante o processo de seleção de técnicas de teste em projetos de software. O planejamento e resultados deste estudo estão publicados em (DIAS-NETO e TRAVASSOS, 2009c)

A hipótese geral definida para este experimento está descrita a seguir de forma inicial, pois na continuidade deste capítulo esta será detalhada para cada aspecto examinado durante o experimento.

- **Hipótese Nula (H_0):** Não existe diferença em usar *Porantim* e o Esquema de Caracterização no processo de seleção de TTBMs em projetos de software.

- **Hipótese Alternativa (H_1):** O processo de seleção apresenta melhores resultados usando *Porantim* se comparado com o uso do *esquema de caracterização* para todos os projetos.
- **Hipótese Alternativa (H_2):** O processo de seleção apresenta melhores resultados usando o *esquema de caracterização* se comparado com o uso de *Porantim* para todos os projetos.
- **Hipótese Alternativa (H_3):** O processo de seleção apresenta melhores resultados para alguns projetos se compararmos com outros, independentemente da abordagem usada, ou seja, cada projeto pode possibilitar melhorias no processo de seleção a partir de suas características.
- **Hipótese Alternativa (H_4):** Dependendo do projeto de software, o processo de seleção apresenta melhores resultados usando *Porantim* ou o *esquema de caracterização*.

A hipótese nula será dividida em sub-hipóteses nulas para cada aspecto examinado durante o experimento. Os aspectos que serão avaliados sobre *Porantim* durante o experimento são: *Completude*, *Efetividade*, *Eficiência*, *Usabilidade* e *Satisfação do Usuário*. O aspecto *Satisfação do Usuário* não será considerado para o propósito de estabelecer uma hipótese que possa ser refutada por meio de evidências estatísticas. Este aspecto será avaliado, de forma qualitativa, examinando as opiniões de cada participante, mais que por meio de uma análise estatística rigorosa.

A próxima Seção descreve o planejamento deste estudo, indicando as variáveis a serem observadas para cada aspecto avaliado.

5.2 Planejamento do Estudo

5.2.1 Definição das Hipóteses

Colocar as hipóteses à prova envolve verificar se existe uma dependência entre o processo de seleção, a abordagem usada para a seleção de TTBMs e o projeto para o qual a seleção é feita. No entanto, melhorar é um termo genérico que requer explicação adicional. Por esse motivo, a hipótese genérica será dividida em cinco hipóteses mais detalhadas, cada uma delas se referindo a um aspecto a ser avaliado no estudo, e suas hipóteses alternativas associadas:

- **Para Completude:**

- **Hipótese Nula 1 (H_{01}):** A completude das informações originais para realizar a seleção é independente da abordagem usada e do projeto considerado.
- **Hipótese Alternativa (H_{11}):** *Porantim* provê um conjunto de informações mais completo para todos os projetos.
- **Hipótese Alternativa (H_{21}):** O *esquema de caracterização* provê um conjunto de informações mais completo para todos os projetos.
- **Hipótese Alternativa (H_{31}):** O projeto considerado é que faz o conjunto original de informações ser mais ou menos completo.
- **Hipótese Alternativa (H_{41}):** Dependendo do projeto considerado, o *esquema de caracterização* ou *Porantim* irá prover um conjunto de informações original mais completo.

- **Para Efetividade:**

- **Hipótese Nula 1 (H_{02}):** A efetividade do processo de seleção é independente da abordagem usada e do projeto considerado.
- **Hipótese Alternativa (H_{12}):** *Porantim* melhora a efetividade da seleção para todos os projetos.
- **Hipótese Alternativa (H_{22}):** O *esquema de caracterização* melhora a efetividade da seleção para todos os projetos.
- **Hipótese Alternativa (H_{32}):** É o projeto considerado que melhora ou piora a efetividade da seleção.
- **Hipótese Alternativa (H_{42}):** Dependendo do projeto considerado, a melhor efetividade será atingida usando o *esquema de caracterização* ou *Porantim*.

- **Para Eficiência:**

- **Hipótese Nula 1 (H_{03}):** A eficiência do processo de seleção é independente da abordagem usada e do projeto considerado.
- **Hipótese Alternativa (H_{13}):** *Porantim* melhora a eficiência da seleção para todos os projetos.
- **Hipótese Alternativa (H_{23}):** O *esquema de caracterização* melhora a eficiência da seleção para todos os projetos.
- **Hipótese Alternativa (H_{33}):** É o projeto considerado quem melhora ou piora a eficiência da seleção.
- **Hipótese Alternativa (H_{43}):** Dependendo do projeto considerado, a melhor eficiência será atingida usando o *esquema de caracterização* ou *Porantim*.

- **Para Usabilidade:**

- **Hipótese Nula 1 (H_{04}):** A usabilidade do processo de seleção é independente da abordagem usada e do projeto considerado.
- **Hipótese Alternativa (H_{14}):** A usabilidade de *Porantim* é melhor que a usabilidade do *esquema de caracterização* para todos os projetos.
- **Hipótese Alternativa (H_{24}):** A usabilidade do *esquema de caracterização* é melhor que a usabilidade de *Porantim* para todos os projetos.
- **Hipótese Alternativa (H_{34}):** É o projeto considerado que melhora ou piora a usabilidade do processo de seleção.
- **Hipótese Alternativa (H_{44}):** Dependendo do projeto considerado, a usabilidade é melhor para o esquema de caracterização ou *Porantim*.

5.2.2 Medições

Q1: Qual é a completude da abordagem de seleção?

Métrica 1.1: informações disponíveis durante a seleção.

Q2: Qual é a efetividade da abordagem de seleção?

Métrica 2.1: *grau de adequação* das TTBMs selecionadas para os projetos de software.

Q3: Qual é a eficiência da abordagem de seleção?

Métrica 3.1: *tempo* gasto na seleção.

Métrica 3.2: *recursos* gastos na seleção.

Q4: Qual é a usabilidade da abordagem de seleção?

Métrica 4.1: *problemas* encontrados durante a seleção.

Existe uma variedade de possíveis soluções quando estudamos o relacionamento entre processo de seleção, abordagem adotada e projeto de software.

- A abordagem de seleção é significante para o processo de seleção. Neste caso, é necessário estudar qual abordagem se comporta melhor e tentar obter explicações sobre as razões pelas quais isso acontece.
- O projeto que está sendo considerado é significante para o processo de seleção. Neste caso, será necessário estudar quais projetos se comportam melhor e tentar obter explicações sobre as razões pelas quais isso acontece.
- A combinação do projeto que está sendo considerado e a abordagem de seleção é significante para o processo de seleção. Neste caso, é necessário estudar qual abordagem se comporta melhor para quais projetos e tentar obter explicações sobre as razões pelas quais isso acontece.
- Nem a abordagem de seleção nem, o projeto que está sendo considerado e nem a combinação dos dois é significante para o processo de seleção. Neste caso, é necessário dar explicações das razões pelas quais isso acontece.

Para satisfação do usuário, os participantes irão avaliar quão satisfeitos eles estão com *Porantim*.

5.2.3 Parâmetros

Os parâmetros (características do projeto que permanecem inalteradas ao longo do experimento) são discutidos a seguir:

- **Experiência dos Participantes:** participantes com diferentes graus de experiência podem ser considerados, provendo a possibilidade de estudar se participantes diferentes respondem igualmente à utilização de *Porantim*. No entanto, a única população disponível no momento é composta por estudantes de graduação e pós-graduação, que compõem a categoria de participantes inexperientes.
- **Tarefa a ser realizada:** os participantes podem ser selocitados para realizarem diferentes tarefas, variando desde uma mera seleção de TTBMs a ser usada em projetos, passando pela preparação de um plano de teste até a execução dos testes. Pelas razões de tempo e característica dos participantes, foi decidido que a tarefa a ser realizada seria restrita à simples seleção de TTBMs. Isso possibilita avaliar o desempenho de *Porantim* com respeito à sua função principal.
- **Documentação:** neste experimento, foi decidido fornecer aos participantes caracterizações de projetos, principalmente porque como eles irão

selecionar, mas não aplicar as TTBMs, eles não precisam usar qualquer outro documento técnico do projeto.

5.2.4 Fatores e Suas Alternativas

As variáveis cujos valores serão alterados são: a abordagem de seleção e o projeto de software.

5.2.4.1 Abordagem de Seleção

O fator **abordagem de seleção** terá três possíveis alternativas: *esquema de caracterização*, uma versão incompleta e uma completa de *Porantim*.

- **Seleção usando o esquema de caracterização:** os participantes receberam o repositório com 13 TTBMs que está descrito no APÊNDICE B, organizado de acordo com os atributos de caracterização usados pela abordagem proposta por VEGAS e BASILI (2005), na qual eles irão usar para decidir quais TTBMs eles selecionariam para os projetos de software aos quais foram alocados.
- **Seleção usando a versão incompleta de *Porantim*:** os participantes receberam o mesmo repositório descrito no APÊNDICE B organizado de acordo com os novos atributos de caracterização providos por *Porantim*, sem os indicadores de grau de adequação gerados por esta abordagem, e a partir disso, eles deverão selecionar TTBMs para os projetos aos quais foram alocados. Esta versão foi usada para avaliar se apenas os novos atributos de caracterização de TTBMs sugeridos por *Porantim* trariam melhorias significativas ao processo de seleção de TTBM.
- **Seleção usando a versão completa de *Porantim*:** os participantes receberam o mesmo repositório descrito no APÊNDICE B organizado de acordo com a versão completa de *Porantim*, com os novos atributos de caracterização de TTBMs e indicador de grau de adequação usado por *Porantim*, e a partir disso eles deverão selecionar TTBMs para os projetos aos quais foram alocados. Esta versão foi usada para avaliar a influência de todas as funcionalidades de *Porantim* no processo de seleção de TTBMs.

5.2.4.2 Projeto de Software

O fator **projeto de software** inclui todas as características do projeto, incluindo não apenas o software a ser desenvolvido, mas também o orçamento e restrições de tempo do projeto, características da equipe, etc.

Neste experimento, foram selecionados quatro projetos de software com diferentes perfis e características, os mesmos usados no estudo original que foi seguido nesta repetição (VEGAS e BASILI, 2005). Segundo os autores do estudo original, apesar de não cobrir todas as possíveis categorias de projetos de software, eles tentam representar diferentes cenários de desenvolvimento de software, consistindo em um termo científico chamado de fator aleatório (VEGAS e BASILI, 2005). Os participantes recebem o documento de caracterização do projeto de software, para cada projeto de software selecionado, contendo informações que eles precisam sobre o software a ser testado. Os quatro projetos de software são:

- **Gerenciamento de Vídeo-Locadora (V):** Sistema de Informação baseado em banco de dados – o objetivo é construir um sistema de gerenciamento de uma vídeo-locadora (Tabela 5.1).

Tabela 5.1. Caracterização do Projeto Vídeo-Locadora

Atributos de Caracterização	Valor
Características do Software/Projeto a ser desenvolvido	
Plataforma de execução do software	Sistema de Informação
Paradigma de desenvolvimento adotado no projeto	Orientado a Objetos
Linguagem de programação utilizada para construir o software	Java
Modelo comportamental/estrutural de software provido pelo projeto	Diagrama de Casos de Uso Diagrama de Atividades Diagrama de Estado Diagrama de Classes
Tecnologia usada para modelagem do software	UML
Duração estimada para o projeto	2 meses para os testes (tempo suficiente)
Complexidade do problema	Baixa
Freqüência de mudança dos requisitos	Baixa
Tamanho estimado da aplicação	Pequena
Características desejadas para os testes a serem realizados	
Nível(is) de Teste desejado(s) para o projeto	Teste de Sistema
Tipo de Técnica de Teste a ser aplicado	Funcional
Características de qualidade do software definidas nos requisitos e que devem ser avaliadas ao longo do projeto	Funcionalidade Eficiência (Tempo de Resposta)
Tipos de Falhas que desejam ser reveladas	Código Faltando Erros de Navegação Erro de interface Erros de Banco de Dados Lógica Incorreta no Código Tipo de dados incorreto
Custo desejado para os testes	A empresa possui verba para adquirir ferramentas
Ferramentas para realização dos testes	A empresa não possui ferramentas

- **Controle de Empréstimo Bancário (B):** Sistema batch – o objetivo é construir um software para aprovação e rejeição de empréstimos bancários e calcular os valores a serem pagos no empréstimo (Tabela 5.2).

Tabela 5.2. Caracterização do Projeto Controle de Empréstimo Bancário

Atributos de Caracterização	Valor
Características do Software/Projeto a ser desenvolvido	
Plataforma de execução do software	Sistema batch – Concorrente
Paradigma de desenvolvimento adotado no projeto	Baseado em especificação formal
Linguagem de programação utilizada para construir o software	Linguagem de programação C
Modelo comportamental/estrutural de software provido pelo projeto	Máquina de Estado Finito
Tecnologia usada para modelagem do software	Diagrama de Estado UML
Duração estimada para o projeto	1 mês para os testes (tempo bastante apertado)
Complexidade do problema	Alta (muitas fórmulas)
Freqüência de mudança dos requisitos	Alta (de acordo com a legislação)
Tamanho estimado da aplicação	Pequeno
Características desejadas para os testes a serem realizados	
Nível(is) de Teste desejado(s) para o projeto	Teste de Unidade
Tipo de Técnica de Teste a ser aplicado	Estrutural
Características de qualidade do software definidas nos requisitos e que devem ser avaliadas ao longo do projeto	Funcionalidade Segurança
Tipos de Falhas que desejam ser reveladas	Erro de arquivo de dados Erro de Shell Script Erro de exceção
Custo desejado para os testes	A empresa possui verba para adquirir ferramentas
Ferramentas para realização dos testes	A empresa não possui ferramentas

- **Estacionamento (E):** sistema síncrono – o objetivo é construir um software para controlar a disponibilidade de vagas em estacionamento de carros (Tabela 5.3).

Tabela 5.3. Caracterização do Projeto Estacionamento

Atributos de Caracterização	Valor
Características do Software/Projeto a ser desenvolvido	
Plataforma de execução do software	Síncrono
Paradigma de desenvolvimento adotado no projeto	Sistema Concorrente
Linguagem de programação utilizada para construir o software	Linguagem de programação CC-Modula ou ADA
Modelo comportamental/estrutural de software provido pelo projeto	Descrição do Ambiente
Tecnologia usada para modelagem do software	LUSTRE
Duração estimada para o projeto	2 meses para os testes (tempo suficiente)
Complexidade do problema	Baixa
Freqüência de mudança dos requisitos	Baixa
Tamanho estimado da aplicação	Pequena
Características desejadas para os testes a serem realizados	
Nível(is) de Teste desejado(s) para o projeto	Teste de Sistema Teste de Integração
Tipo de Técnica de Teste a ser aplicado	Funcional
Características de qualidade do software definidas nos requisitos e que devem ser avaliadas ao longo do projeto	Funcionalidade Segurança Eficiência (Tempo de resposta)
Tipos de Falhas que desejam ser reveladas	Código faltando Erro de Navegação Erro de exceção

	Lógica Incorreta no código Tipo de dados incorreto
Custo desejado para os testes	A empresa não está interessada em comprar ferramentas
Ferramentas para realização dos testes	A empresa não possui ferramentas

- **Controle de Nível de Água em Represa (R):** sistema de tempo real – o objetivo é construir um software para monitorar o nível de água em uma represa (Tabela 5.4).

Tabela 5.4. Caracterização do Projeto Controle de Nível de Água em Represa

Atributos de Caracterização	Valor
Características do Software/Projeto a ser desenvolvido	
Plataforma de execução do software	Tempo Real
Paradigma de desenvolvimento adotado no projeto	Sistema de tempo real
Linguagem de programação utilizada para construir o software	Ada
Modelo comportamental/estrutural de software provido pelo projeto	Especificação do software através de modelos Formais Diagrama de Estado Diagrama de Seqüência/Colaboração
Tecnologia usada para modelagem do software	Modelos UML são providos pela equipe de desenvolvimento
Duração estimada para o projeto	Prazo apertado
Complexidade do problema	Média
Freqüência de mudança dos requisitos	Alta
Tamanho estimado da aplicação	Média
Características desejadas para os testes a serem realizados	
Nível(is) de Teste desejado(s) para o projeto	Teste de Unidade
Tipo de Técnica de Teste a ser aplicado	Estrutural
Características de qualidade do software definidas nos requisitos e que devem ser avaliadas ao longo do projeto	Funcionalidade Segurança Eficiência
Tipos de Falhas que desejam ser reveladas	Código faltando Erro de Interface Erro de arquivo de dados Erro de Shell Script Erro de exceção Lógica Incorreta no código
Custo desejado para os testes	A empresa não está interessada em comprar ferramentas
Ferramentas para realização dos testes	A empresa possui ferramentas adequadas

5.2.5 Técnicas de Teste Baseado em Modelos utilizadas no Experimento

Ao total, 13 TTBMs caracterizadas a partir das 3 abordagens de seleção de TTBMs utilizadas neste estudo foram disponibilizadas. Este número representa exatamente o mesmo número de técnicas de teste disponibilizadas no estudo original

publicado por VEGAS e BASILI (2005). A lista de TTBMs adotadas neste estudo está apresentada na Tabela 5.5. A caracterização detalhada de cada TTBM está descrita no Apêndice B.

Tabela 5.5. TTBMs disponibilizadas no estudo em Represa

Identificador	Referência
Técnica #1	(ABDURAZIK e OFFUT, 2000)
Técnica #2	(BRIAND e LABICHE, 2002)
Técnica #3	(CHANG *et al*., 1996)
Técnica #4	(HARTMANN e NAGIN, 2000)
Técnica #5	(KIM *et al*., 1999)
Técnica #6	(FRIEDMAN *et al*., 2002)
Técnica #7	(ALAGAR *et al.*, 2000)
Técnica #8	(STOBIE, 2005)
Técnica #9	(VIEIRA *et al.*, 2006)
Técnica #10	(du BOUSQUET *et al.*, 1999)
Técnica #11	(CHUNG *et al.*1999)
Técnica #12	(PARISSIS e VASSEY, 2003)
Técnica #13	(DALAL *et al.*, 1999)

Para cada projeto de software, foram selecionadas 2 TTBMs como escolhas "corretas" para o projeto, ou seja, aquelas que mais se adéquam ao projeto de software. Essas escolhas foram realizadas por 2 especialistas em TBM antes da execução deste estudo. A Tabela 5.6 apresenta a lista de TTBMs "corretas" para cada projeto de software.

Tabela 5.6. Oráculo de TTBMs "corretas" por Projeto de Software

Projeto de Software	TTBM "corretas"
Vídeo Locadora	• Técnica #2 (BRIAND e LABICHE, 2002) • Técnica #9 (VIEIRA *et a*l., 2006)
Empréstimo	• Técnica #3 (CHANG *et al*., 1999) • Técnica #6 (FRIEDMAN *et a*l., 2002)
Estacionamento	• Técnica #10 (du BOUSQUET *et al*., 1999) • Técnica #12 (PARISSIS e VASSEY, 2003)
Represa	• Técnica #1 (ABDURAZIK e OFFUT, 2000) • Técnica #7 (ALAGAR *et a*l., 2000)

Além disso, foram calculados os graus de adequação de cada TTBM para cada projeto utilizando a estratégia *Porantim*. Estes indicadores estão apresentados na Tabela 5.7. Esses valores apenas serão apresentados para os participantes que usaram a versão completa de *Porantim*.

Tabela 5.7. Oráculo de TTBMs "corretas" por Projeto de Software

TTBM	Projeto de Software							
	Vídeo Locadora		Empréstimo		Estacionamento		Represa	
	Grau	Ordem	Grau	Ordem	Grau	Ordem	Grau	Ordem
Abordagem 1	52,47%	4	34,59%	10	36,49%	10	56,69%	2
Abordagem 2	67,89%	2	33,74%	13	39,44%	6	35,80%	12
Abordagem 3	31,68%	10	46,27%	3	36,71%	8	47,86%	3
Abordagem 4	61,62%	3	45,16%	4	45,16%	5	39,85%	7
Abordagem 5	38,07%	6	50,19%	2	20,60%	12	45,55%	5
Abordagem 6	31,51%	12	58,84%	1	31,29%	11	46,22%	4
Abordagem 7	37,50%	7	34,91%	9	38,13%	7	56,79%	1
Abordagem 8	37,05%	9	36,88%	6	36,65%	9	23,35%	13
Abordagem 9	91,45%	1	34,57%	11	45,25%	4	35,88%	11
Abordagem 10	31,68%	11	35,43%	8	65,43%	1	37,49%	10
Abordagem 11	15,83%	13	38,69%	5	18,38%	13	39,69%	8
Abordagem 12	37,27%	8	36,28%	7	65,43%	2	38,89%	9
Abordagem 13	44,43%	5	34,44%	12	48,52%	3	41,42%	6

5.2.6 Variáveis de Resposta

As variáveis de resposta a serem consideradas estão listadas na Tabela 5.8. Uma série de formulários foi usada para apoiar na seleção de TTBMs. Eles foram re-aproveitados totalmente do estudo original (VEGAS e BASILI, 2005) e estão apresentados no Apêndice B deste trabalho. Como um dos objetivos do estudo foi conduzir uma replicação externa de outro estudo realizado, todas as questões do estudo original foram aplicadas para possibilitar uma futura comparação dos resutados.

Tabela 5.8. Questões Variáveis de Resposta do Experimento

Aspecto	Variáveis de Resposta
Completude	• Quantas informações foram usadas para seleção de TTBMs? • Conjunto de informações usadas para seleção • Quantas informações estão faltando para apoiar a seleção? • Conjunto de informações que estão faltando para apoiar a seleção
Efetividade	• Quantas TTBMs foram consideradas durante a seleção? • Quantas TTBMs foram selecionadas? • Quais são as TTBMs selecionadas?

Eficiência	• Tempo gasto estudando as TTBMs • Tempo de seleção • Tempo gasto consultando a descrição dos atributos de caracterização usados por *Porantim*
Usabilidade	• Quantos problemas foram encontrados durante a seleção? • Descrição dos problemas encontrados
Satisfação do Usuário	• Pontos positivos e negativos do uso de *Porantim* • Você estaria preparado para usar novamente *Porantim*? • Melhorias que você faria em *Porantim* • O que você gostou? E o que você não gostou? • *Porantim* mudou a forma como você observa o problema de seleção de TTBMs? • O que você aprendeu com o uso de *Porantim*? • Você realizaria a seleção de TTBMs de forma diferente em uma próxima vez? • Adequabilidade dos nomes e organização de *Porantim*
Caracterização do participante	• Experiência de trabalho • Experiência em Teste de Software • Você normalmente seleciona TTBMs? • Quais os tipos de informações que você acredita que precisaria? • Qual o conjunto de problemas que você acredita que irá encontrar? • Quanto tempo você acredita que levaria para fazer a seleção de TTBMs?

5.3 Projeto do Estudo

O objetivo primário deste experimento é encontrar qual a influência que *Porantim* possui no processo de seleção de TTBMs em um projeto de software. Ele também visa a examinar as diferenças com respeito ao uso de outra abordagem de seleção que não seja *Porantim* (neste caso, o esquema de caracterização). Para isso, decidiu-se dividir a população em quatro grupos, cada um fará a seleção duas vezes, de acordo com o padrão exibido na Tabela 5.9.

Tabela 5.9. Questões Alocação das abordagens de seleção por grupos

	Grupo 1	**Grupo 2**	**Grupo 3**	**Grupo 4**
Seleção 1	*Porantim* (completa)	Esquema	Esquema	*Porantim* (incompleta)
Seleção 2	*Porantim* (completa)	*Porantim* (incompleta)	Esquema	Esquema

Esta distribuição possui a seguinte razão:

- Os Grupos 2 e 4 são apropriados para estudar o efeito da abordagem de seleção.
- Os Grupos 1 e 3 são apropriados para avaliar o efeito de aprendizagem (se a segunda seleção é melhor que a primeira e até que ponto).

- Além disso, o Grupo 1 é apropriado para avaliar se o uso completo de *Porantim* traz benefícios relacionados aos aspectos avaliados no experimento, sem introduzir viés de comparar o resultado do uso completo de *Porantim* com o *Esquema de Caracterização*.

Ainda pretende-se avaliar se os diferentes tipos de participantes (dependendo de qual grupo eles pertencem) estão satisfeitos com a(s) abordagem(ns) que eles usaram para seleção de TTBMs e quais eles preferem.

A respeito do tamanho, cada grupo conteve uma quantidade similar de membros (a diferença será no máximo de um membro entre os grupos, caso o total de participantes não seja um valor múltiplo de 4).

Para avaliar outro aspecto, os participantes de todos os grupos, após concluir a etapa de seleção de TTBMs, deverão preencher um formulário indicando em que ordem as TTBMs deveriam ser selecionadas para o projeto que estão tratando, da mais relevante para a menos relevante (Formulário E8 – Apêndice B).

5.3.1 Projeto de Dois-Fatores com Replicação

Cada participante deve fazer duas seleções. Como existem quatro projetos diferentes, será possível distribuir os projetos aleatoriamente entre as diferentes seleções (1 e 2) e diferentes grupos (1 a 4). A Tabela 5.10 exibe os subgrupos (“A” a “M”) que estabelece a relação entre os grupos e a alocação de projetos, e a Tabela 5.11 exibe a alocação dos participantes por subgrupo (grupos x alocação de projetos).

Tabela 5.10. Alocação de projetos por subgrupos

REPLICAÇÃO	Seleção 1				Seleção 2			
	(V)	(B)	(E)	(R)	(V)	(B)	(E)	(R)
A	X						X	
B	X							X
C		X					X	
D		X						X
E			X		X			
F			X			X		
G				X	X			
H				X		X		
I	X					X		
J		X			X			
L			X					X
M				X			X	

Tabela 5.11. Alocação de participantes por grupos e subgrupos

	Grupo 1	Grupo 2	Grupo 3	Grupo 4
A	P1	P26	P39	P16
B	P17	P42	P11	P32
C	P9	P18	P31	P44
D	P37	P2	P15	P28
E	P41	P30	P19	P12
F	P25	P14	P3	P40
G	P13	P38	P27	P4
H	P29	P10	P43	P20
I	P33	P6	P23	P48
J	P21	P34	P47	P8
L	P5	P46	P35	P24
M	P45	P22	P7	P36

5.3.2 Procedimento Experimental

O experimento foi organizado em 3 sessões, como mostrado na Tabela 5.12.

Tabela 5.12. Procedimento Experimental

	Sessão 1	Sessão 2	Sessão 3
Grupo 1	Sessão de Explicação sobre TBM, o Estudo e apresentação do esquema e *Porantim*	Seleção com *Porantim*	Seleção com *Porantim*
Grupo 2		Seleção com Esquema	Seleção com *Porantim*
Grupo 3		Seleção com Esquema	Seleção com Esquema
Grupo 4		Seleção com *Porantim*	Seleção com Esquema

5.3.3 Ameaças à Validade

5.3.3.1 Ameaças Internas

- **Cópia:** existe a possibilidade dos estudantes copiarem os resultados de outros, pois o exercício é realizado fora do horário da aula sem qualquer monitoração. Para tentar minimizar este problema, os estudantes serão avisados que eles serão avaliados não com base nas técnicas selecionadas, mas no esforço em fazer o exercício. Além disso, eles devem ser relembrados que estão realizando o exercício voluntariamente.
- **Capacidade:** é fato que os participantes não possuem a mesma habilidade para resolução de problemas. Para tentar minimizar este efeito, os participantes serão alocados aos grupos de forma aleatória.
- **Estrutura das abordagens de seleção:** a abordagem *Porantim*, por definição, contém uma série de facilidades adicionais não previstas no esquema de caracterização para apoiar a seleção de TTBMs, tais como os

atributos de caracterização específicos para TTBMs e indicadores de grau de adequação das técnicas em relação a um projeto de software. Para tentar minimizar esse problema, os participantes que forem usar as duas estratégias só receberão uma versão limitada contendo apenas a caracterização das TTBMs com o intuito de aproximá-la ao máximo da outra abordagem usada. Apenas os participantes que forem usar apenas *Porantim* nas duas etapas do experimento irão utilizar a versão completa da abordagem.

5.3.3.2 Ameaças Externas

- **Experiência:** os participantes não possuem experiência na seleção de TTBMs para projetos de software. Isso pode resultar em vantagens para alguma das estratégias. Sendo assim, o resultado não pode ser generalizado para todos os tipos de participantes.
- **Projetos:** quatro diferentes projetos de software estão sendo usados com o objetivo de tertar prover uma representação da realidade. No entanto, experimentos com outros tipos de projetos devem ser executados, assim como nem todas as situações possíveis de um projeto de software podem ser representadas.
- **TTBM** usadas: um conjunto de TTBMs que corresponde a um conjunto o mais variado possível (com diferentes categorias de software, níveis de teste, linguagens de programação, dentre outras informações) está sendo usado. Novos estudos com um novo conjunto de TTBMs devem ser executados.

5.3.4 Análise dos Dados

Será utilizado o método estatístico Análise de Variância (ANOVA) para realizar a análise dos dados coletados durante o experimento. ANOVA será usada para estudar o relacionamento entre uma variável de resposta quantitativa e um ou mais fator qualitativo. Seu objetivo será determinar se a diferença entre as médias das variáveis de resposta nos grupos estabelecidas pela combinação de níveis de fatores é estatisticamente significante. Este foi o mesmo método estatístico usado no estudo original conduzido por VEGAS e BASILI (2005). Para apoiar na execução de ANOVA neste estudo, foi utilizada a ferramenta JMP 4[12].

[12] JMP é um software de apoio à análise de dados estatísticos (http://www.jmp.com/).

5.4 Execução do Estudo

Este estudo experimental foi executado em 2 (duas) diferentes instâncias: a primeira instância foi realizada com alunos de pós-graduação e a segunda, com alunos de graduação, como será descrito nas próximas subseções.

5.4.1 Execução da Rodada POS – Estudantes de Pós-Graduação

A primeira instância, que será chamada de POS, foi executada com 21 (vinte e um) estudantes de pós-graduação (mestrado e doutorado) de uma disciplina de engenharia de software realizada na COPPE (Instituto Alberto Luiz Coimbra de Pós Graduação). Eles foram identificados com as IDs de P01 até P22, de acordo com a combinação do grupo ao qual pertence e os projetos que foram alocados, conforme apresentado na Tabela 5.11.

Tais estudantes relataram possuir experiência em Engenharia de Software e Teste de Software, porém pouca ou nenhuma experiência na seleção de técnicas de teste para projetos de software, conforme apresentado na Tabela 5.13 que descreve a caracterização dos participantes da instância POS. Durante a execução do estudo, 1 (um) participante da instância POS (P04) desistiu do estudo, e portanto suas respostas não serão analisadas (este participante está com sua linha destacada na Tabela 5.13).

Tabela 5.13. Caracterização dos Participantes da instância POS

Grupo	ID	Experiência em ES	Experiência em Testes	Experiência em TBM	Experiência na seleção de técnicas de teste
1	P01	5	5	1	2
	P05	5	5	1	1
	P09	3	3	2	2
	P13	4	2	2	2
	P17	4	4	1	1
	P21	3	2	1	1
2	P02	5	5	2	2
	P06	5	3	1	3
	P10	5	5	1	4
	P14	5	2	2	2
	P18	5	5	1	5
	P22	5	4	1	1
3	P03	5	5	1	3
	P07	3	3	1	1
	P11	4	2	2	2
	P15	5	3	1	2
	P19	4	3	1	1
4	**P04**	**5**	**4**	**2**	**2**
	P08	3	2	1	1
	P12	5	5	2	3
	P16	5	2	2	2
	P20	5	1	1	1

LEGENDA:
1 = nenhuma
2 = estudei em aula ou livro
3 = pratiquei em 1 projeto em sala de aula
4 = usei em 1 projeto na indústria
5 = usei em vários projetos na indústria

5.4.2 Execução da Rodada GRAD – Estudantes de Graduação

A segunda instância, que será chamada de GRAD, foi executada com 34 (trinta e quatro) estudantes de graduação de uma disciplina de Engenharia de Software Orientada a Objetos realizada na UFRJ (Universidade Federal do Rio de Janeiro).

Como o objetivo do estudo era avaliar todas as possíveis combinações de abordagens de seleção e projetos de software, os participantes foram identificados com as IDs a partir de P21 até P48 (para cobrir as combinações não usadas com a instância POS) e depois foi reiniciada a distribuição dos partipantes da ID P01 até P06 (totalizando 34 participante), seguindo a distribuição apresentada na Tabela 5.11.

Tais estudantes relataram possuir pouca ou nenhuma experiência em Engenharia de Software e Teste de Software e nenhuma experiência na seleção de técnicas de teste para projetos de software, conforme apresentado na Tabela 5.14 que descreve a caracterização dos participantes da instância GRAD. Durante a execução do estudo, 4 (quatro) participantes da instância GRAD (P22, P28, P37, P46) desistiram do estudo, e, portanto, suas respostas não serão analisadas (estes participantes estão com sua linha destacada na Tabela 5.14).

Tabela 5.14. Caracterização dos Participantes da instância GRAD

Grupo	ID	Experiência em ES	Experiência em Testes	Experiência em TBM	Experiência na seleção de técnicas de teste
1	P01	3	2	1	1
	P05	4	4	1	1
	P21	3	2	1	1
	P25	3	2	1	1
	P29	3	2	1	1
	P33	4	2	1	1
	P37	**5**	**4**	**1**	**1**
	P41	3	2	1	1
	P45	3	2	1	
2	P02	3	2	1	1
	P06	3	2	1	1
	P22	**3**	**2**	**1**	**1**
	P26	3	2	1	1
	P30	3	2	1	1
	P34	3	2	1	1
	P38	3	2	1	1
	P42	3	2	1	1
	P46	**3**	**2**	**1**	**1**

3	P03	3	2	1	1
	P23	3	2	1	1
	P27	4	2	1	1
	P31	3	2	1	1
	P35	3	2	1	1
	P39	3	2	1	1
	P43	3	2	1	1
	P47	4	2	1	1
4	P04	3	2	1	1
	P24	3	2	1	1
	P28	**3**	**2**	**1**	**1**
	P32	3	2	1	1
	P36	3	2	1	1
	P40	3	2	1	1
	P44	3	2	1	1
	P48	3	2	1	1
	LEGENDA: 1 = nenhuma 2 = estudei em aula ou livro 3 = pratiquei em 1 projeto em sala de aula 4 = usei em 1 projeto na indústria 5 = usei em vários projetos na indústria				

5.5 Análise Quantitativa dos Resultados do Estudo

Após a execução do estudo em suas 2 (duas) instâncias, foi realizada a fase de análise dos dados seguindo os procedimentos definidos durante o planejamento do estudo. As variáveis analisadas durante este estudo estão descritas na Tabela 5.15 juntamente com sua descrição.

Tabela 5.15. Variáveis avaliadas no Estudo Experimental

Aspecto	Variáveis analisadas	Descrição
Completude	Percentual de informações (atributos) usadas na seleção	Avalia a proporção de atributos usados para apoiar a seleção de TTBMs, analisando se todo o conjunto tem sido efetivamente utilizado
	Quantidade de informações faltando durante a seleção	Analisa em cada abordagem de seleção quais os atributos que estariam faltando para caracterização de uma TTBM e que influenciariam no processo de seleção
Efetividade	Percentual de escolhas corretas	Analisa se as abordagens direcionam à seleção de TTBMs adequadas a um projeto, de acordo com um oráculo pré-definido por especialistas em TBM
Eficiência	Tempo de seleção (em minutos)	Analisa se as abordagens possibilitam à seleção de TTBMs em um tempo adequado
	Percentual de escolhas corretas / tempo de seleção	Analisa se as abordagens direcionam à seleção de TTBMs adequadas a um projeto utilizando um pequeno conjunto de atributos de caracterização durante o processo
	Percentual de escolhas corretas / Percentual de informações usadas	Analisa se as abordagens direcionam à seleção de TTBMs adequadas a um projeto em um tempo reduzido
Usabilidade	Quantidade de problemas/dúvidas reportadas durante a seleção	Analisa os problemas relatados durante o uso de cada abordagem de seleção

Independentemente da instância, o primeiro passo realizado para todas as variáveis foi a análise de *outlier*[13]. Após isso, foram realizadas as análises quantitativa e qualitativa para cada variável, conforme descrito nas subseções seguintes. Os dados de cada instância serão analisados separadamente.

5.5.1 Análise Quantitativa dos Resultados da Rodada POS

A análise dos dados será descrita a seguir com os resultados obtidos pela instância POS para cada variável avaliada neste estudo:

5.5.1.1 Completude – Percentual de Informações (Atributos) usadas (usados)

- **Análise de *Outlier***

Aplicando-se análise de *outlier*, observa-se que nenhum participante é avaliado como sendo um *outlier* na avaliação das respostas por abordagem de seleção e por projeto de software.

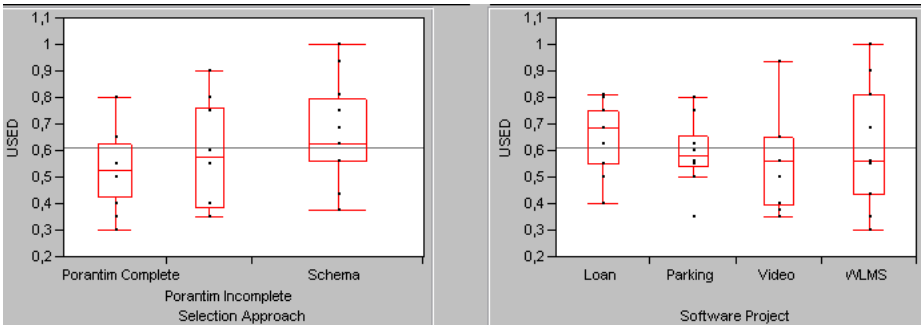


Figura 5.1. Análise de *Outlier* – Completude (Percentual de Informações usadas)

- **Análise da Média**

Analisando a média entre as abordagens de seleção, observa-se que existe uma diferença significativa entre as médias obtidas pela versão completa de *Porantim* (52,50% dos atributos são usados para seleção) em comparação com sua versão incompleta (60,00% dos atributos) e principalmente com o esquema de caracterização (68,87% dos atributos). Isso indica que *Porantim* possibilita a seleção de TTBMs com uma quantidade menor de atributos a serem analisados. Analisando a média entre os projetos de software, percebemos que não existe uma diferença significativa de

[13] Em Estatística, um *outlier* é uma observação que está numericamente distante quando comparada aos demais dados de um conjunto observado.

médias (55,50%; 59,12%; 63,29%; 65,90%), o que indica que os projetos caracterizados não influenciam no percentual de informações a serem usadas para se tomar a decisão sobre a seleção de TTBMs. Tais resultados estão descritos na Tabela 5.16 e Figura 5.2.

Tabela 5.16. Média e Desvio Padrão – Completude (Percentual de Informações usadas)

Fator	Alternativa	Média	Desvio Padrão
Abordagem de Seleção	EC[14]	68,87%	15,58%
	PI[16]	60,00%	19,57%
	PC[16]	52,50%	13,73%
Projeto de Software	(V)	55,50%	17,19%
	(B)	65,90%	13,25%
	(E)	59,12%	12,55%
	(R)	63,29%	22,83%

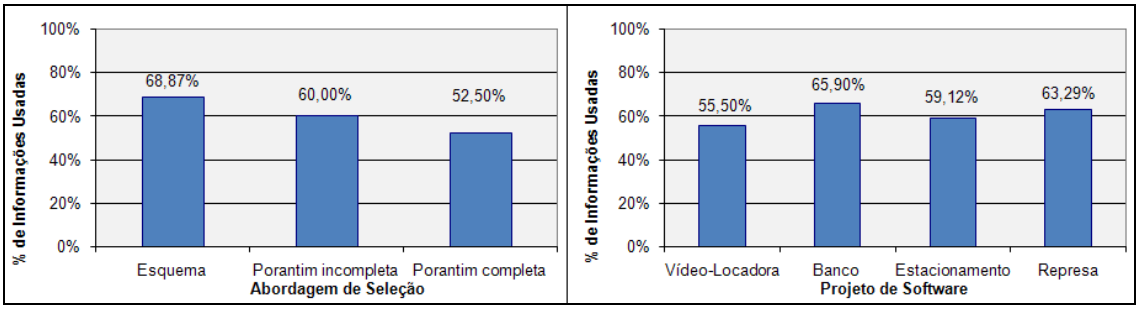


Figura 5.2. Média do Percentual de Informações usadas por Abordagem de Seleção e Projeto de Software

- **Análise de Variância – ANOVA**

Aplicando-se testes estatísticos, percebe-se que, apesar do valor médio de percentual de informações usadas ser menor para a versão completa de *Porantim*, não existe diferença estatística entre as abordagens. No entanto, os valores obtidos estão quase na fronteira, como pode ser observado nos círculos representando cada abordagem (o círculo de *Porantim* completa está quase sem interseção com o círculo do esquema de caracterização) e pelo *p-value*[15] que ficou em 0,0608, próximo ao limite de 0,05 que representa o valor de significância estatística[16] (taxa de erro) de 5% (ver lado esquerdo da Figura 5.3).

[14] EC = Esquema de Caracterização; PI = *Porantim* incompleta; PC = *Porantim* Completa.

[15] *P-value* é a probabilidade de que nossa amostra podia ter sido tirada de uma população sendo testada assumindo que a hipótese nula seja verdadeira. Se o valor é menor que o valor de significância (ver item 15), indica que a hipótese nula é falsa. Caso contrário, a hipótese nula não pode ser rejeitada.

[16] Valor de Significância de um teste é a probabilidade máxima de rejeitar acidentalmente uma hipótese nula verdadeira (uma decisão conhecida como erro de tipo I). O nível de significância de um resultado é também chamado de α.

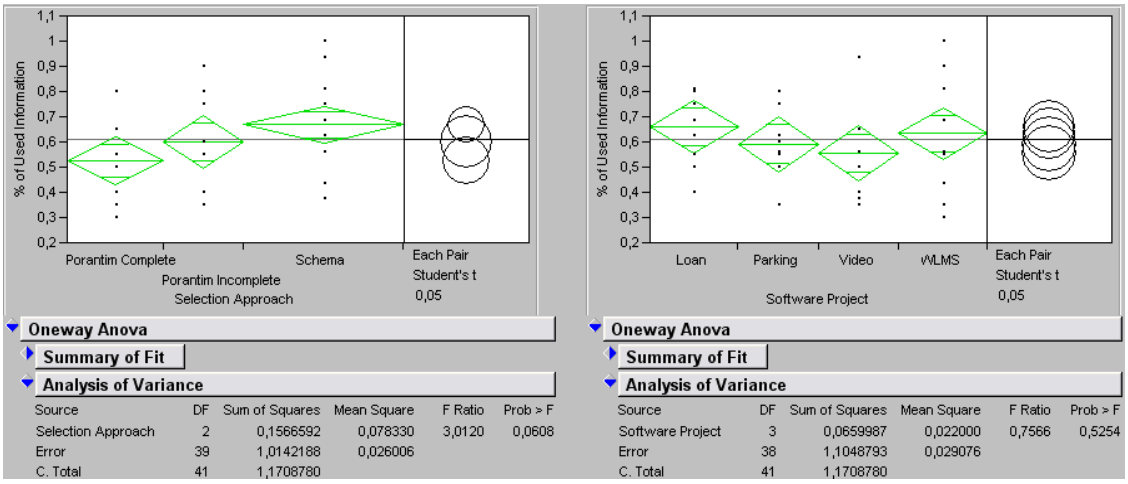


Figura 5.3. ANOVA para Completude (Percentual de Informações usadas)

Fazendo a mesma análise para avaliar o impacto do projeto de software no percentual de informações usadas, percebe-se que não existe diferença estatística entre eles, como pode ser observado pela interseção entre todos os círculos representando cada projeto de software e pelo *p-value* que ficou em 0,5254, superior ao valor 0,05 que representa o valor de significância de 5% (ver lado direito da Figura 5.3).

Sendo assim, os resultados indicam que nenhum dos fatores investigados (abordagem de seleção e projeto de software) possui influência na redução do percentual de informações usadas para seleção de TTBMs em projetos de software.

5.5.1.2 Completude – Quantidade de Informações Faltando

- **Análise de *Outlier***

Aplicando-se análise de *outlier*, observa-se que um participante é avaliado como sendo um *outlier* na avaliação das respostas por abordagem de seleção e por projeto de software, pois relatou a ausência de 5 informações (Figura 5.4). Com isso, seus resultados foram removidos da análise desta variável (Figura 5.5).

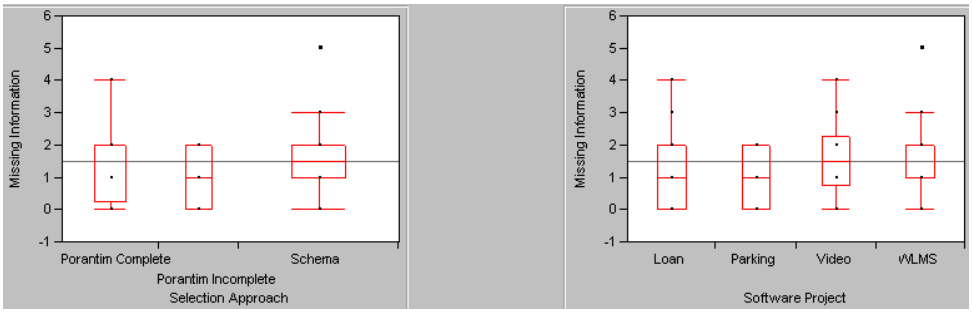


Figura 5.4. Análise de *Outlier* – Completude (Quantidade de Informações faltando) – Antes da Exclusão

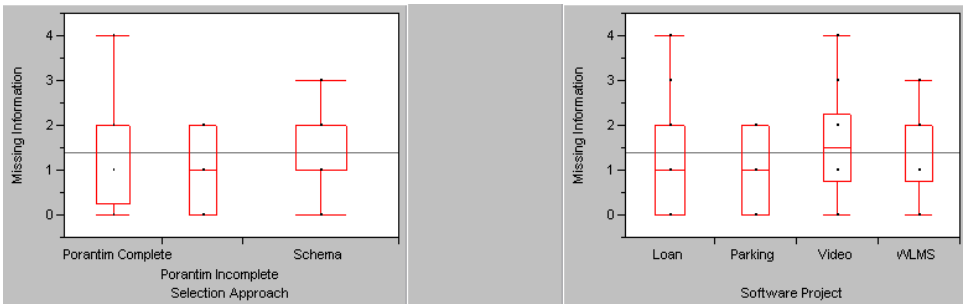


Figura 5.5. Análise de *Outlier* – Completude (Quantidade de Informações faltando) – Após a remoção do *Outlier*

- **Análise da Média**

Analisando a média entre as abordagens de seleção, observa-se que existe uma diferença significativa entre as médias obtidas pela versão completa de *Porantim* (1,66) e do Esquema de Caracterização (1,42) em relação à média obtida na versão incompleta de *Porantim* (1,00). Analisando a média entre os projetos de software, percebemos que existe uma diferença significativa de valores entre o projeto Estacionamento (1,0) para com os demais projetos (1,45 ; 1,5 ; 1,6), o que indica que os projetos caracterizados poderiam influenciar de alguma forma no total de informações faltando para se tomar a decisão sobre a seleção de TTBMs. Tais resultados estão descritos na Tabela 5.17 e Figura 5.6.

Tabela 5.17. Média e Desvio Padrão – Completude (Quantidade de Informações faltando)

Fator	Alternativa	Média	Desvio Padrão
Abordagem de Seleção	EC	1,42	1,01
	PI	1,00	0,81
	PC	1,66	1,37
Projeto de Software	(V)	1,60	1,26
	(B)	1,45	1,29
	(E)	1,00	0,81
	(R)	1,50	0,97

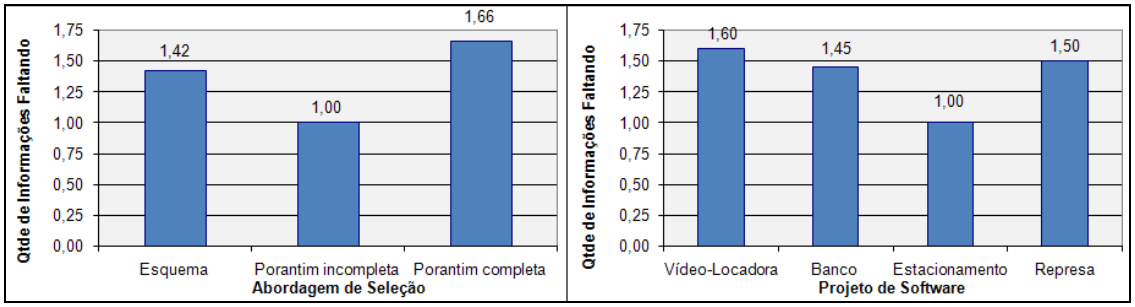


Figura 5.6. Média da Quantidade de Informações faltando por Abordagem de Seleção e Projeto de Software

- **Análise de Variância – ANOVA**

Aplicando-se testes estatísticos, percebe-se que, apesar do valor médio do total de informações faltando ser menor para a versão incompleta de *Porantim*, não existe diferença estatística entre as abordagens, como pode ser observado pela interseção entre todos os círculos representando cada abordagem de seleção e pelo *p-value* que ficou em 0,3664, superior ao valor 0,05 que representa o valor de significância estatística (taxa de erro) de 5% (ver lado esquerdo da Figura 5.7).

Fazendo a mesma análise para avaliar o impacto do projeto de software no total de informações faltando, percebe-se que não existe diferença estatística entre eles, como pode ser observado pela interseção entre todos os círculos representando cada projeto de software e pelo *p-value* que ficou em 0,6348, superior ao valor 0,05 que representa o valor de significância estatística (taxa de erro) de 5% (ver lado direito da Figura 5.7).

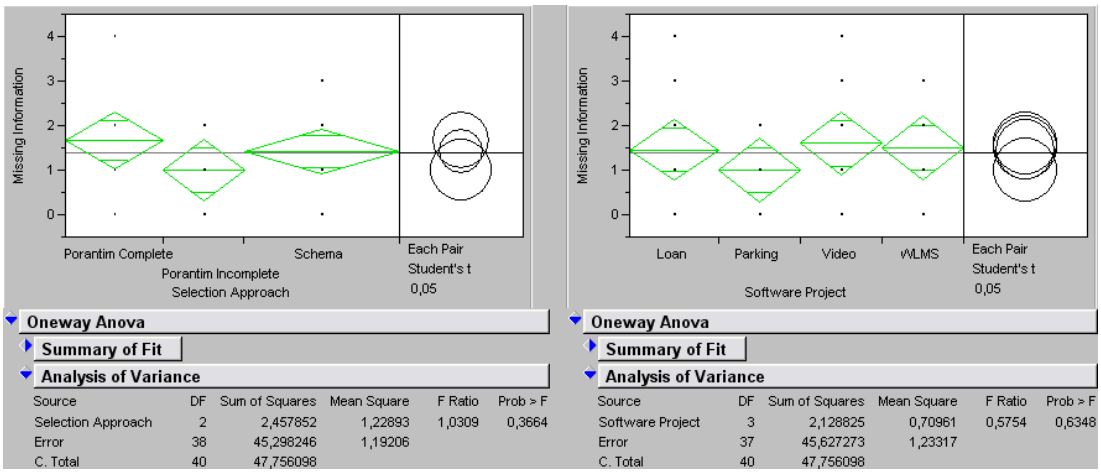


Figura 5.7. ANOVA para Completude (Quantidade de Informações faltando)

Sendo assim, os resultados indicam que nenhum dos fatores investigados (abordagem de seleção e projeto de software) possui influência na redução da quantidade de informações faltando para caracterização de TTBMs.

5.5.1.3 Efetividade – Percentual de Escolhas Corretas

- **Análise de *Outlier***

Aplicando-se análise de *outlier*, observa-se que nenhum participante é avaliado como sendo um *outlier* na avaliação das respostas por abordagem de seleção e por projeto de software (Figura 5.8).

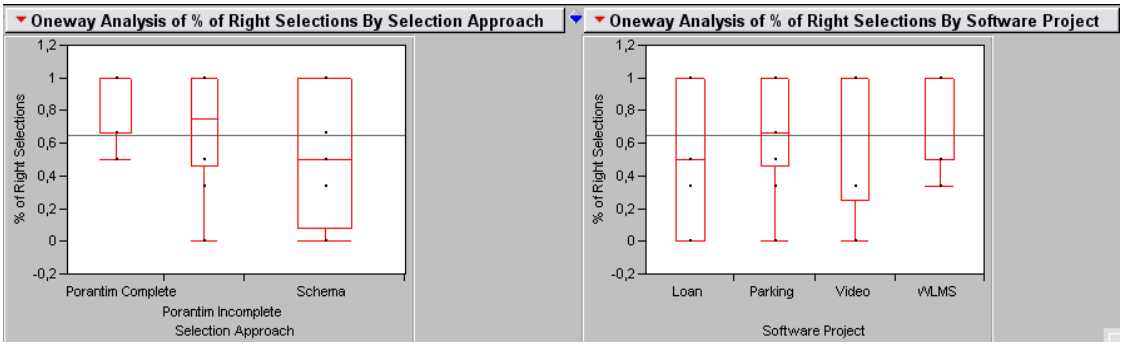


Figura 5.8. Análise de *Outlier* – Efetividade (Percentual de Escolhas Corretas)

- **Análise da Média**

Analisando a média entre as abordagens de seleção, observa-se que existe uma diferença significativa entre as médias obtidas pela versão completa de *Porantim* (86% de acerto) em comparação com sua versão incompleta (68% de acerto) e principalmente com o esquema de caracterização (50% de acerto). Isso indica que a versão completa de *Porantim* apoiaria no aumento do percentual de escolhas corretas de TTBMs em projetos de software. Analisando a média entre os projetos de software, percebemos uma maior proximidade entre elas, apesar da média obtida para os projetos Represa e Empréstimo Bancário serem menores que para os projetos Vídeo e Estacionamento. Tais resultados estão descritos na Tabela 5.18 e Figura 5.9.

Tabela 5.18. Média e Desvio Padrão – Completude (Percentual de Escolhas Corretas)

Fator	Alternativa	Média	Desvio Padrão
Abordagem de Seleção	EC	50,00%	38,99%
	PI	68,33%	36,38%
	PC	86,11%	21,12%
Projeto de Software	(V)	73,33%	43,88%
	(B)	57,57%	44,32%
	(E)	68,33%	33,74%
	(R)	60,60%	26,11%

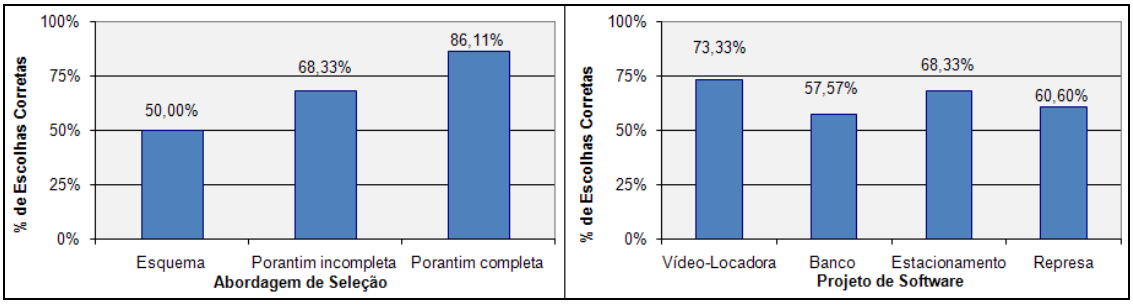


Figura 5.9. Média do Percentual de Escolhas Corretas por Abordagem de Seleção e Projeto de Software

- **Análise de Variância – ANOVA**

Aplicando-se testes estatísticos, percebe-se que existe diferença estatística entre as abordagens *Porantim* Completa e o Esquema de Caracterização, como pode ser observado pela não-interseção entre os círculos representando cada abordagem e pelo *p-value* que ficou em 0,0215, inferior ao valor 0,05 que representa o valor de significância estatística (taxa de erro) de 5% (ver lado esquerdo da Figura 5.10). Percebe-se ainda que a versão incompleta de *Porantim* não possui diferença estatística em relação às outras duas abordagens, o que significa que apenas incrementar o conjunto de atributos de caracterização de TTBMs apesar de reduzir o tempo gasto para seleção, não é suficiente para prover melhorias significativas nessa variável, o que só foi possível a partir da versão completa de *Porantim*, com indicadores e gráficos de apoio à seleção de TTBMs.

Fazendo a mesma análise para avaliar o impacto do projeto de software no percentual de seleções corretas, percebe-se que não existe diferença estatística entre eles, como pode ser observado pela interseção entre todos os círculos representando cada projeto de software e pelo *p-value* que ficou em 0,7682, superior ao valor 0,05 que representa o valor de significância estatística (taxa de erro) de 5% (ver lado direito da Figura 5.10).

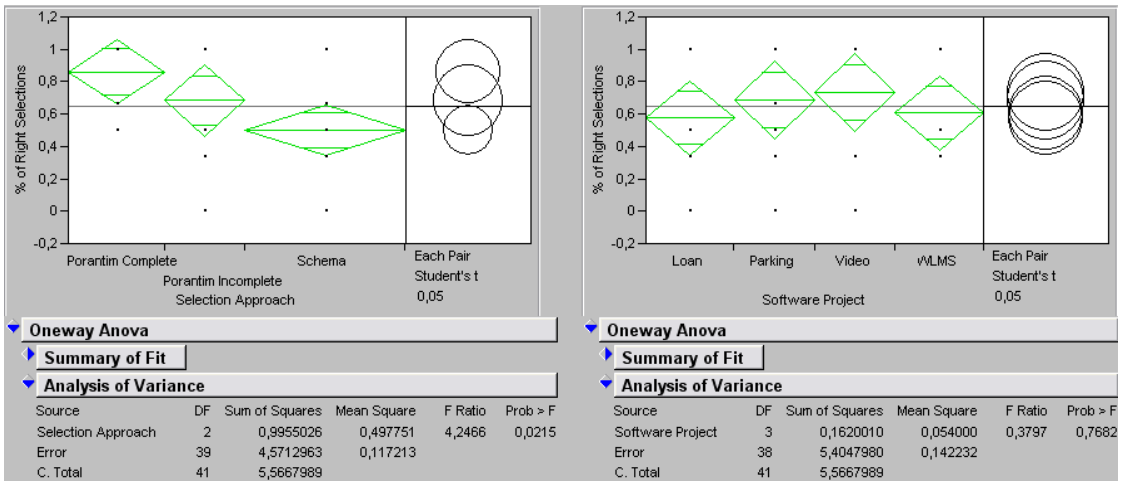


Figura 5.10. ANOVA para Efetividade (Percentual de Escolhas Corretas)

Sendo assim, os resultados indicam que o fator avaliado que influencia no aumento do percentual de seleções de TTBMs corretas em projetos de software seria a abordagem de seleção adotada.

5.5.1.4 Eficiência – Tempo de Resposta

- **Análise de *Outlier***

Aplicando-se análise de *outlier*, observa-se que nenhum participante é avaliado como sendo um *outlier* na avaliação das respostas por abordagem de seleção e por projeto de software (Figura 5.11).

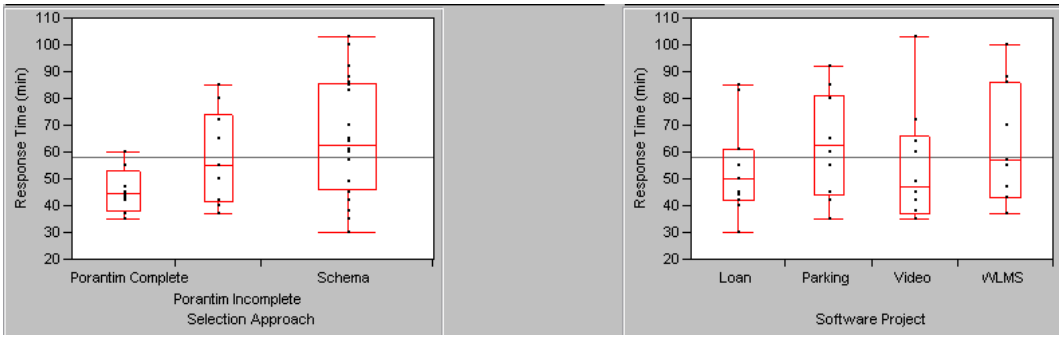


Figura 5.11. Análise de *Outlier* – Eficiência (Tempo de Resposta)

- **Análise da Média**

Analisando a média entre as abordagens de seleção (Tabela 5.19 – Figura 5.12), observa-se que existe uma diferença significativa entre a média obtida pela versão completa de *Porantim* (45,25 min) em comparação com sua versão incompleta (58,10 min) e principalmente com o esquema de caracterização (65,50 min). Isso indica que *Porantim* apoiaria na redução do tempo gasto para a seleção de TTBMs em projetos de software. Analisando a média entre os projetos de software (Tabela 5.19 – Figura 5.12), percebemos uma maior proximidade entre elas, apesar da média obtida para os projetos Represa e Estacionamento serem maiores que para os projetos Vídeo e Empréstimo Bancário.

Tabela 5.19. Média e Desvio Padrão – Eficiência (Tempo de Resposta)

Fator	Alternativa	Média	Desvio Padrão
Abordagem de Seleção	EC	65,50 min	22,1490 min
	PI	58,10 min	16,8684 min
	PC	45,25 min	8,0128 min
Projeto de Software	(V)	54,30 min	21,3128 min
	(B)	53,63 min	17,2129 min
	(E)	62,40 min	18,9748 min
	(R)	61,54 min	21,6627 min

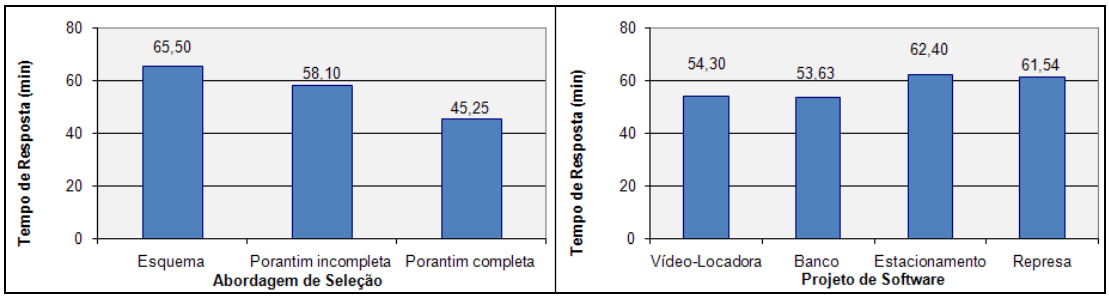


Figura 5.12. Média do Tempo de Resposta por Abordagem de Seleção e Projeto de Software

- **Análise de Variância – ANOVA**

Aplicando-se testes estatísticos, percebe-se que existe diferença estatística entre as abordagens *Porantim* Completa e o Esquema de Caracterização, como pode ser observado pela não-interseção entre os círculos representando cada abordagem e pelo *p-value* que ficou em 0,0141, inferior ao valor 0,05 que representa o valor de significância estatística (taxa de erro) de 5% (ver lado esquerdo da Figura 5.13). Percebe-se ainda que a versão incompleta de *Porantim* não possui diferença estatística em relação às outras duas abordagens, o que significa que apenas incrementar o conjunto de atributos de caracterização de TTBMs apesar de reduzir o tempo gasto para seleção, não é suficiente para prover melhorias significativas nessa variável, o que só foi possível a partir da versão completa de *Porantim*, com indicadores e gráficos de apoio à seleção de TTBMs.

Fazendo a mesma análise para avaliar o impacto do projeto de software no tempo gasto para seleção, percebe-se que não existe diferença estatística entre eles, como pode ser observado pela interseção entre todos os círculos representando cada projeto de software e pelo *p-value* que ficou em 0,6361, superior ao valor 0,05 que representa o valor de significância estatística (taxa de erro) de 5% (lado direito da Figura 5.13).

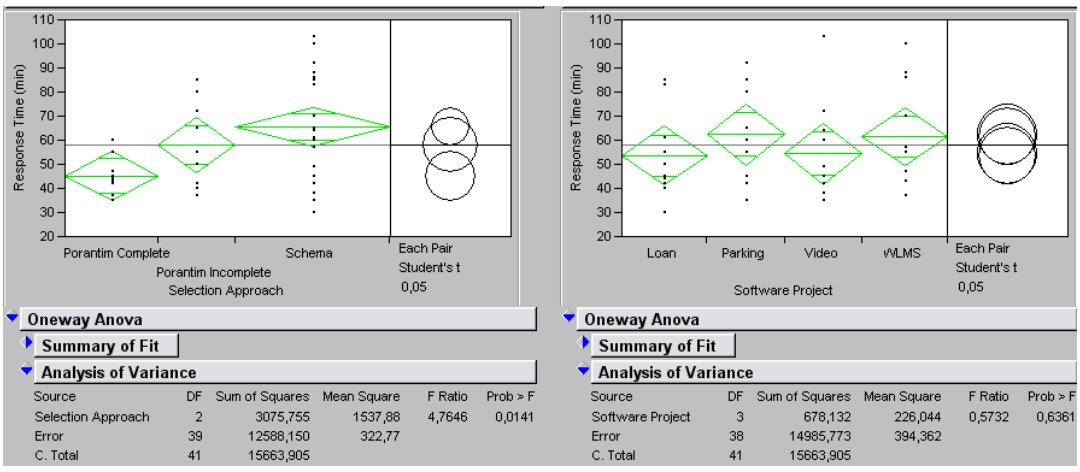


Figura 5.13. ANOVA para Eficiência (Tempo de Resposta)

Sendo assim, os resultados indicam que o fator avaliado que influencia na redução do tempo gasto para seleção de TTBMs em projetos de software seria a abordagem de seleção adotada.

5.5.1.5 Eficiência – Percentual de Escolhas Corretas / Percentual de Informações Usadas

A métrica percentual de escolhas corretas por percentual de informações usadas indica qual o esforço investido por cada abordagem de seleção para se atingir uma seleção mais precisa para um projeto de software. Quanto mais informações usadas para se obter escolhas corretas, menos eficiente seria a abordagem de seleção. Ou seja, quanto maior for este indicador, mais eficiente será a abordagem de seleção.

- **Análise de *Outlier***

Aplicando-se análise de *outlier*, observa-se que nenhum participante é avaliado como sendo um *outlier* na avaliação das respostas por abordagem de seleção e por projeto de software (Figura 5.14).

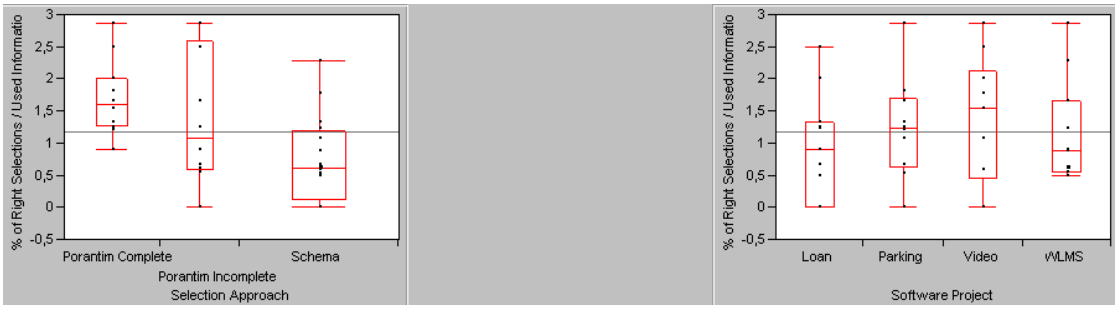


Figura 5.14. Análise de *Outlier* – Eficiência (% de Escolhas Corretas / % de Informações usadas)

- **Análise da Média**

Analisando a média entre as abordagens de seleção, observa-se que existe uma diferença significativa entre as médias obtidas pela versão completa de *Porantim* (1,71) em comparação com sua versão incompleta (1,38) e principalmente com o esquema de caracterização (0,73). Isso indica que a versão completa de *Porantim* possibilita obter um percentual maior de acertos na seleção de TTBMs usando uma quantidade menor de informações/atributos. Analisando a média entre os projetos de software, percebemos um maior proximidade entre elas (0,9 ; 1,14 ; 1,24 ; 1,38), o que indica que o comportamento seria de certa forma similar entre os projetos. Tais resultados estão descritos na Tabela 5.20 e Figura 5.15.

Tabela 5.20. Média e Desvio Padrão – Eficiência (% de Escolhas Corretas / % de Informações usadas)

Fator	Alternativa	Média	Desvio Padrão
Abordagem de Seleção	EC	0,73	0,63
	PI	1,38	1,03
	PC	1,71	0,55
Projeto de Software	(V)	1,38	0,97
	(B)	0,94	0,82
	(E)	1,24	0,78
	(R)	1,14	0,80

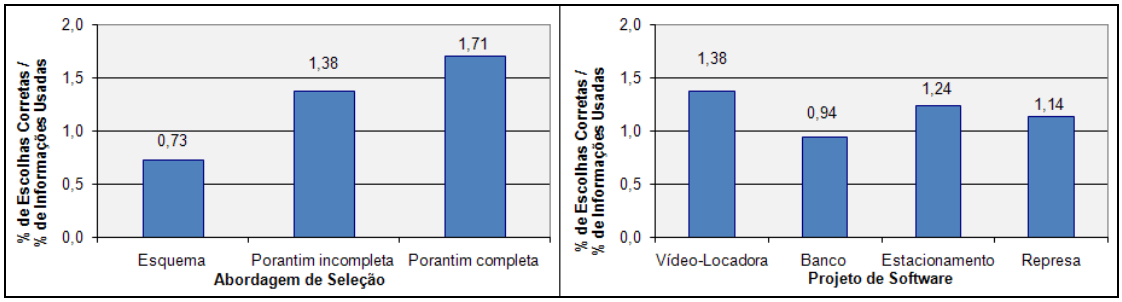


Figura 5.15. Média do "% de Escolhas Corretas / % de Informações usadas" por Abordagem de Seleção e Projeto de Software

- **Análise de Variância – ANOVA**

Aplicando-se testes estatísticos, percebe-se que existe diferença estatística entre as abordagens *Porantim* Completa e o Esquema de Caracterização, como pode ser observado pela não-interseção entre os círculos representando cada abordagem e pelo *p-value* que ficou em 0,0019, quase nulo e bem abaixo do valor 0,05 que representa o valor de significância estatística (taxa de erro) de 5% (ver lado esquerdo da Figura 5.16). Percebe-se ainda uma grande diferença entre a versão incompleta de *Porantim* com sua versão completa, apesar de ainda haver uma interseção entre seus círculos, o que indica que as funcionalidades adicionais providas pela versão completa de *Porantim* contribuem significantemente para tornar a seleção de TTBMs mais eficiente e com o uso de uma quantidade menor de informações.

Fazendo a mesma análise para avaliar o impacto do projeto de software nesta variável, percebe-se que não existe diferença estatística entre eles, como pode ser observado pela interseção entre todos os círculos representando cada projeto de software e pelo *p-value* que ficou em 0,6799, superior ao valor 0,05 que representa o valor de significância estatística (taxa de erro) de 5% (ver lado direito da Figura 5.16).

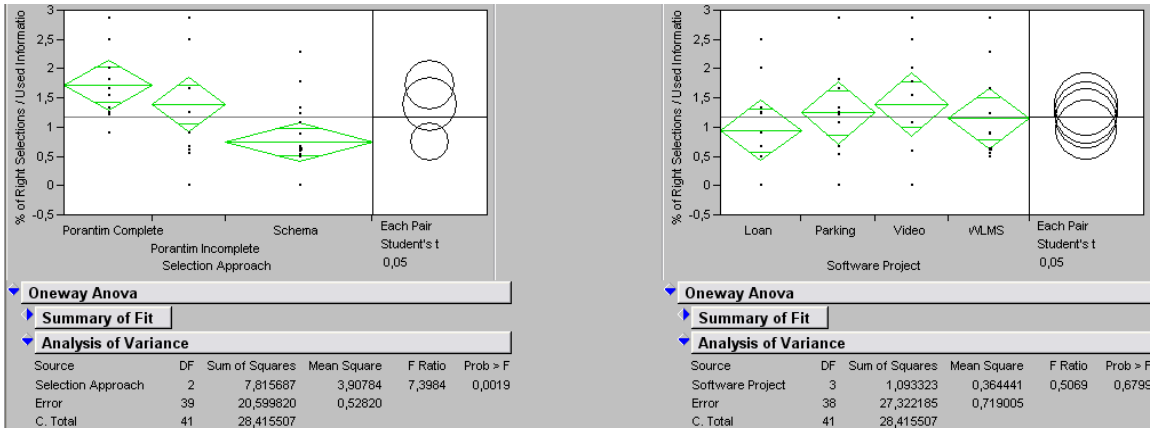


Figura 5.16. ANOVA para Eficiência (% de Escolhas Corretas / % de Informações Usadas)

Sendo assim, os resultados indicam que o fator avaliado que influencia no aumento do percentual de seleções de TTBMs corretas em projetos de software por informações usadas seria a abordagem de seleção adotada.

5.5.1.6 Eficiência – Percentual de Escolhas Corretas / Tempo de Resposta

A métrica percentual de escolhas corretas por tempo de resposta tenta expressar o dinamismo da abordagem de apoio à seleção de TTBMs em indicar um maior percentual de técnicas corretas para um projeto em uma menor quantidade de tempo (medido em minutos). Sendo assim, quanto maior for este valor, mais eficiente será a abordagem de seleção.

- **Análise de *Outlier***

Aplicando-se análise de *outlier*, observa-se que um participante é avaliado como sendo um *outlier* na avaliação das respostas por abordagem de seleção e um outro participante é avaliado como sendo *outlier* por projeto de software (Figura 5.17). Com isso, seus resultados foram removidos da análise desta variável (Figura 5.18).

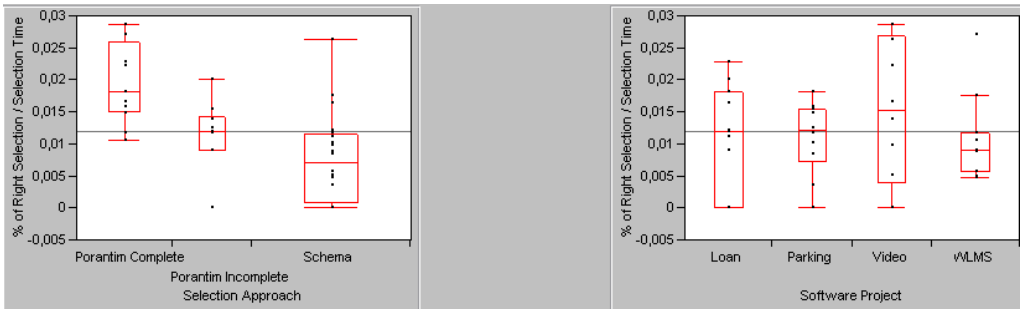


Figura 5.17. Análise de *Outlier* – Eficiência (% de Escolhas Corretas / Tempo de Resposta) – Antes da Exclusão

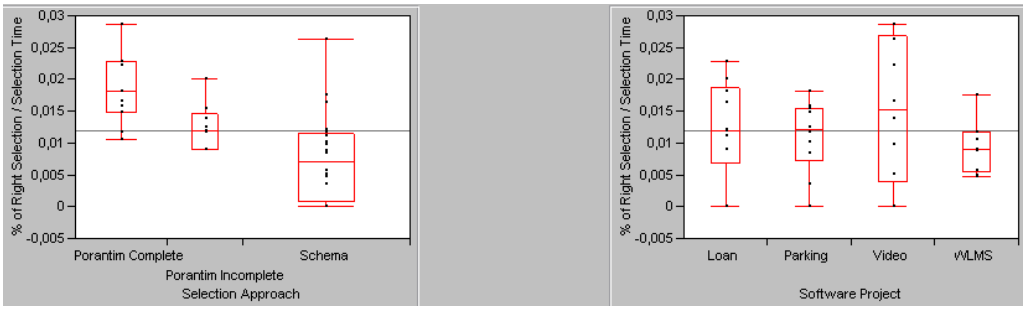


Figura 5.18. Análise de *Outlier* – Eficiência (% de Escolhas Corretas / Tempo de Resposta) – Após a remoção dos *outliers*

- **Análise da Média**

Analisando a média entre as abordagens de seleção, observa-se que existe uma diferença significativa entre as médias obtidas pela versão completa de *Porantim* (1,89% de acertos por minuto) em comparação com sua versão incompleta (1,25% de acertos por minuto) e principalmente com o esquema de caracterização (0,78% de acertos por minuto). Isso indica que a versão completa de *Porantim* possibilita obter um percentual maior percentual de acertos na seleção de TTBMs em uma quantidade mais reduzida de tempo. Analisando a média entre os projetos de software, percebemos que apenas o projeto Vídeo (média = 1,51 % de acerto por minuto) possui um valor de média um pouco diferenciado em relação aos demais projetos (1,21%; 1,10%; 0,93%). Tais resultados estão descritos na Tabela 5.21 e Figura 5.19.

Tabela 5.21. Média e Desvio Padrão – Eficiência (% de Escolhas Corretas / Tempo de Resposta)

Fator	Alternativa	Média	Desvio Padrão
Abordagem de Seleção	EC	0,78 %/min	0,7
	PI	1,25 %/min	0,3
	PC	1,89 %/min	0,6
Projeto de Software	(V)	1,51 %/min	1,11
	(B)	1,21 %/min	0,7
	(E)	1,10 %/min	0,5
	(R)	0,93 %/min	0,6

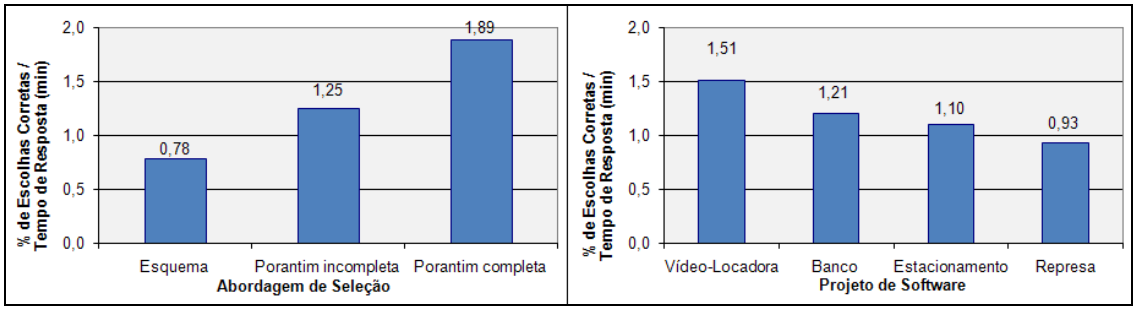


Figura 5.19. Média do "% de Escolhas Corretas / Tempo de Resposta" por Abordagem de Seleção e Projeto de Software

- **Análise de Variância – ANOVA**

Aplicando-se testes estatísticos, percebe-se que existe diferença estatística entre as abordagens *Porantim* Completa em relação às outras duas abordagens avaliadas, como pode ser observado pela não-interseção entre o círculo representando a abordagem *Porantim* completa e o círculo representando o Esquema de Caracterização, além do *p-value* que ficou abaixo de 0,0001, quase nulo (ver lado esquerdo da Figura 5.20). Percebe-se ainda uma diferença entre a versão incompleta de *Porantim* em relação ao esquema de caracterização, porém seus círculos possuem interseção. Este resultado indica que as funcionalidades adicionais providas pela versão completa de *Porantim* contribuem significantemente para tornar a seleção de TTBMs mais eficiente em relação ao percentual de acertos por unidade de tempo.

Fazendo a mesma análise para avaliar o impacto do projeto de software nesta variável, percebe-se que não existe diferença estatística entre eles, como pode ser observado pela interseção entre todos os círculos representando cada projeto de software (apesar de um maior distanciamento do círculo representando o projeto Vídeo) e pelo *p-value* que ficou em 0,4026, superior ao valor 0,05 que representa o valor de significância estatística (taxa de erro) de 5% (ver lado direito da Figura 5.20).

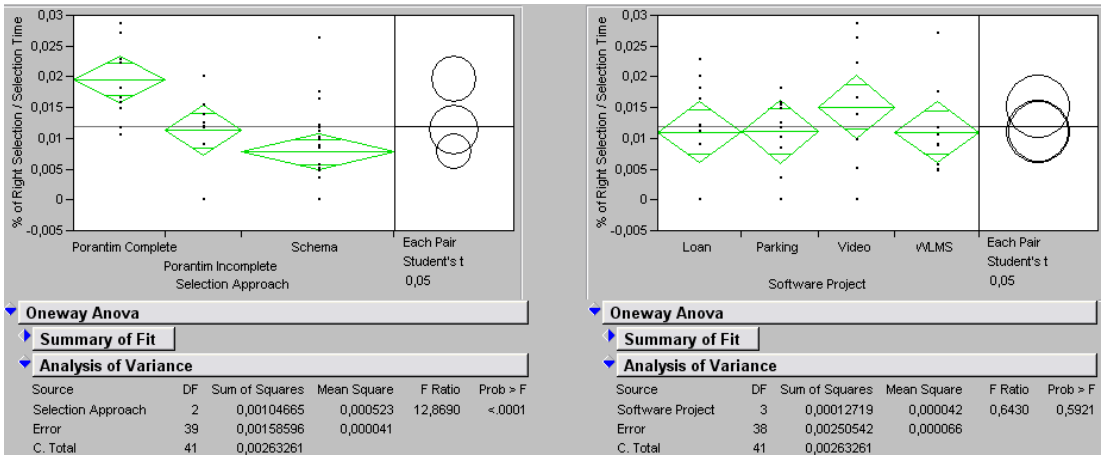


Figura 5.20. ANOVA para Eficiência (% de Escolhas Corretas / Tempo de Resposta)

Sendo assim, os resultados indicam que o fator avaliado que influencia no aumento do percentual de seleções de TTBMs corretas em projetos de software por unidade de tempo seria a abordagem de seleção adotada.

5.5.1.7 Usabilidade – Quantidade de Problemas relatados durante a seleção

- **Análise de *Outlier***

Aplicando-se análise de *outlier*, observa-se que existem 4 *outliers* entre os participantes: 3 associados à abordagem de seleção e 2 associados ao projeto de software, sendo que um deles é um dos *outliers* já identificado para a abordagem de seleção (Figura 5.21). Com isso, seus resultados foram removidos da análise desta variável Figura 5.22).

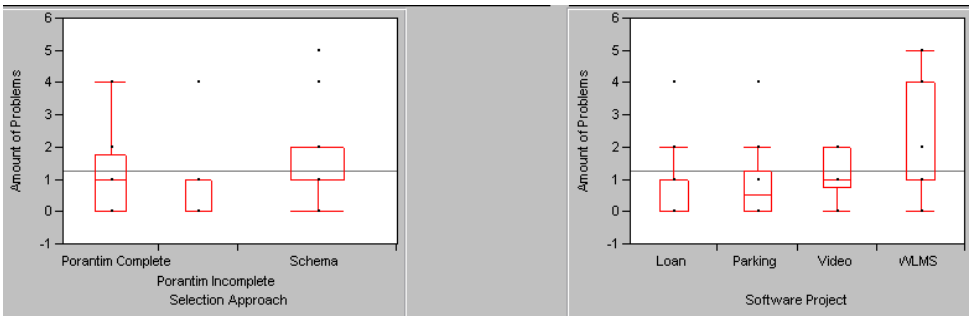


Figura 5.21. Análise de *Outlier* – Usabilidade (Quantidade de Problemas) – Antes da Exclusão

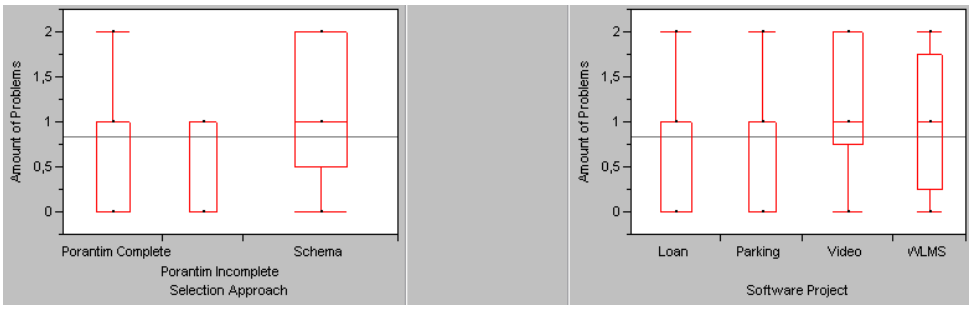


Figura 5.22. Análise de *Outlier* – Usabilidade (Quantidade de Problemas) – Após a remoção do *Outlier*

- **Análise da Média**

Analisando a média entre as abordagens de seleção, observa-se que existe uma diferença significativa entre as médias obtidas pelas versões completa e incompleta de *Porantim* (0,55 e 0,72) em comparação com o esquema de caracterização (1,05). Isso indica que os participantes que usaram *Porantim* relataram menos problemas ou dificuldades em selecionar TTBMs. Analisando a média entre os projetos de software, percebemos uma maior proximidade entre os projetos Estacionamento e Banco (0,55 e 0,67) em relação aos projetos Controle de Represa e Vídeo (1,00 e 1,10), o que indica que as características dos projetos de software poderiam ser um fator que influenciasse na quantidade de problemas relatados durante a seleção de TTBMs. Isso porque o projeto Controle de Represa é dentre os

quatro projetos o que possui maior complexidade em relação ao seu domínio e suas características. Tais resultados estão descritos na Tabela 5.22 e Figura 5.23.

Tabela 5.22. Média e Desvio Padrão – Usabilidade (Quantidade de Problemas)

Fator	Alternativa	Média	Desvio Padrão
Abordagem de Seleção	EC	1,42	1,01
	PI	1,00	0,81
	PC	1,66	1,37
Projeto de Software	(V)	1,60	1,26
	(B)	1,45	1,29
	(E)	1,50	0,81
	(R)	1,50	0,97

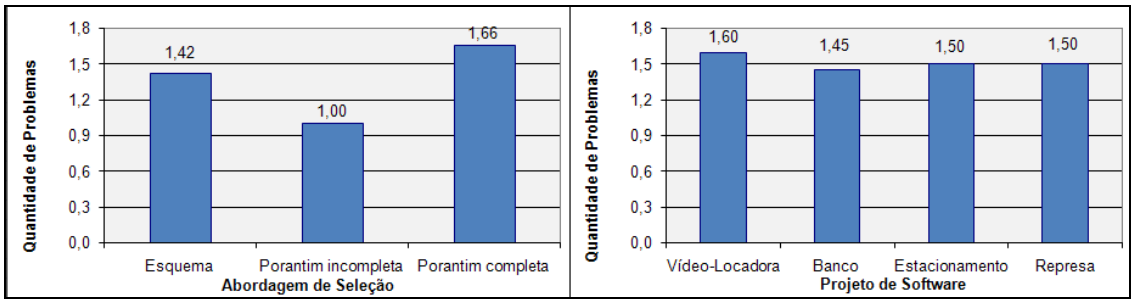


Figura 5.23. Média da Quantidade de Problemas por Abordagem de Seleção e Projeto de Software

- **Análise de Variância – ANOVA**

Aplicando-se testes estatísticos, percebe-se que apesar dos valores de média obtidos por *Porantim* serem inferiores, não existe diferença estatística entre as abordagens de seleção, como pode ser observado pela interseção entre os círculos representando cada abordagem e pelo *p-value* que ficou em 0,2074, acima do valor 0,05 que representa o valor de significância estatística (taxa de erro) de 5% (ver lado esquerdo da Figura 5.24). Percebe-se ainda que os valores obtidos entre as versões incompletas e completas de *Porantim* são quase os mesmos, e os círculos que descrevem cada abordagem estão quase sobrepostos.

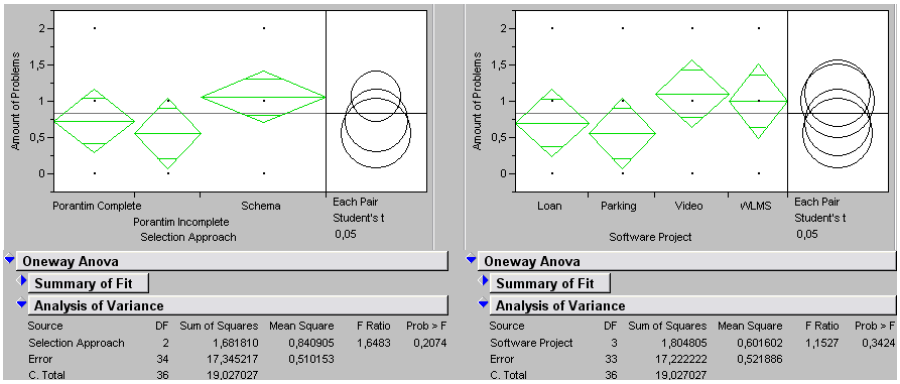


Figura 5.24. ANOVA para Usabilidade (Quantidade de Problemas)

Fazendo a mesma análise para avaliar o impacto do projeto de software nesta variável, percebe-se que não existe diferença estatística entre eles, apesar da grande diferença da média obtida pelo projeto Controle de Represas em relação aos demais. Isso pode ser observado pela interseção entre todos os círculos representando cada projeto de software e pelo *p-value* que ficou em 0,3424, superior ao valor 0,05 que representa o valor de significância estatística (taxa de erro) de 5% (ver lado direito da Figura 5.24).

Sendo assim, os resultados indicam que nenhum dos fatores investigados (abordagem de seleção e projeto de software) possui influência na redução ou aumento da quantidade de problemas relatados durante a seleção de TTBMs.

5.5.2 Análise Quantitativa dos Resultados da Rodada GRAD

A análise dos dados será descrita a seguir com os resultados obtidos pela instância POS para cada variável avaliada neste estudo:

5.5.2.1 Completude – Percentual de Informações (Atributos) usadas (usados)

- **Análise de *Outlier***

Aplicando-se análise de *outlier*, observa-se que 2 (dois) participantes são avaliados como sendo *outliers* na avaliação das respostas por abordagem de seleção (Figura 5.25). Com isso, seus resultados foram removidos da análise desta variável (Figura 5.26).

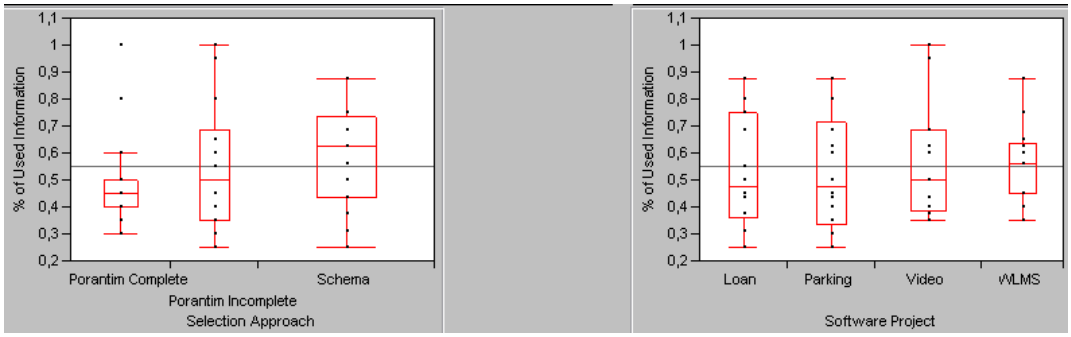


Figura 5.25. Análise de *Outlier* – Completude (Percentual de Informações usadas) – Antes da Exclusão

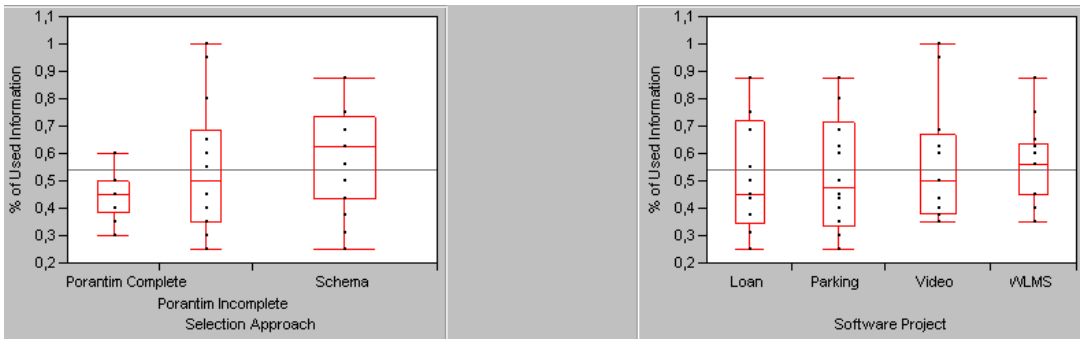


Figura 5.26. Análise de *Outlier* – Completude (Percentual de Informações usadas) – Após a remoção do *Outlier*

- **Análise da Média**

Analisando a média entre as abordagens de seleção, observa-se que existe uma diferença não muito significativa entre as médias obtidas pela versão completa de *Porantim* (46,66% dos atributos são usados para seleção) em comparação com sua versão incompleta (55% dos atributos) e com o esquema de caracterização (58,25% dos atributos). Isso sugere que *Porantim* possibilita a seleção de TTBMs consultando um percentual menor de atributos. Analisando a média entre os projetos de software, percebemos que não existe uma diferença significativa de valores (53,12%; 53,21%; 54,84%; 56,53%), o que indica que as características de um projeto não influenciam no percentual de informações a serem usadas para se tomar a decisão sobre a seleção de TTBMs. Tais resultados estão descritos na Tabela 5.23 e Figura 5.27.

Tabela 5.23. Média e Desvio Padrão – Completude (Percentual de Informações usadas)

Fator	Alternativa	Média	Desvio Padrão
Abordagem de Seleção	EC	58,25%	19,17%
	PI	55,00%	23,45%
	PC	46,66%	7,80%
Projeto de Software	(V)	54,84%	20,08%
	(B)	53,12%	20,13%
	(E)	53,21%	21,14%
	(R)	56,53%	14,69%

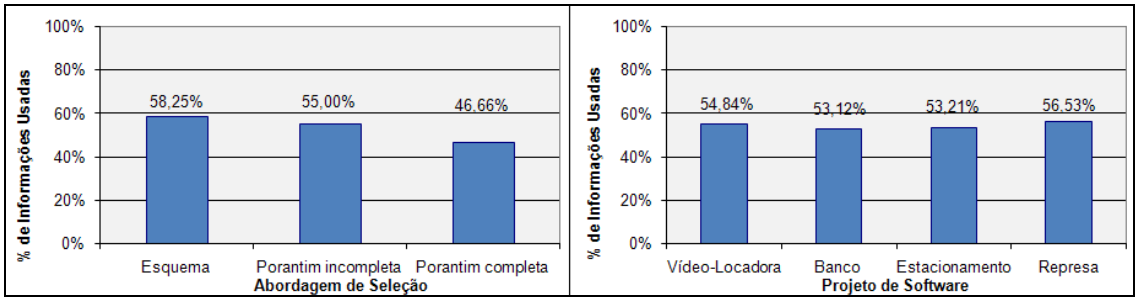


Figura 5.27. Média do Percentual de Informações usadas por Abordagem de Seleção e Projeto de Software

- **Análise de Variância – ANOVA**

Aplicando-se testes estatísticos, percebe-se que, apesar do valor médio de percentual de informações usadas ser menor para a versão completa de *Porantim*, não existe diferença estatística entre as abordagens, como pode ser observado pela interseção entre todos os círculos representando cada abordagem e pelo *p-value* que ficou em 0,1642, superior ao limite de 0,05 que representa o valor de significância estatística (taxa de erro) de 5% (ver lado esquerdo da Figura 5.28).

Fazendo a mesma análise para avaliar o impacto do projeto de software no percentual de informações usadas, percebe-se que não existe diferença estatística entre eles, como pode ser observado pela interseção entre todos os círculos representando cada projeto de software (eles estão quase sobrepostos) e pelo *p-value* que ficou em 0,9644, superior ao valor 0,05 que representa o valor de significância estatística (taxa de erro) de 5% (ver lado direito da Figura 5.28).

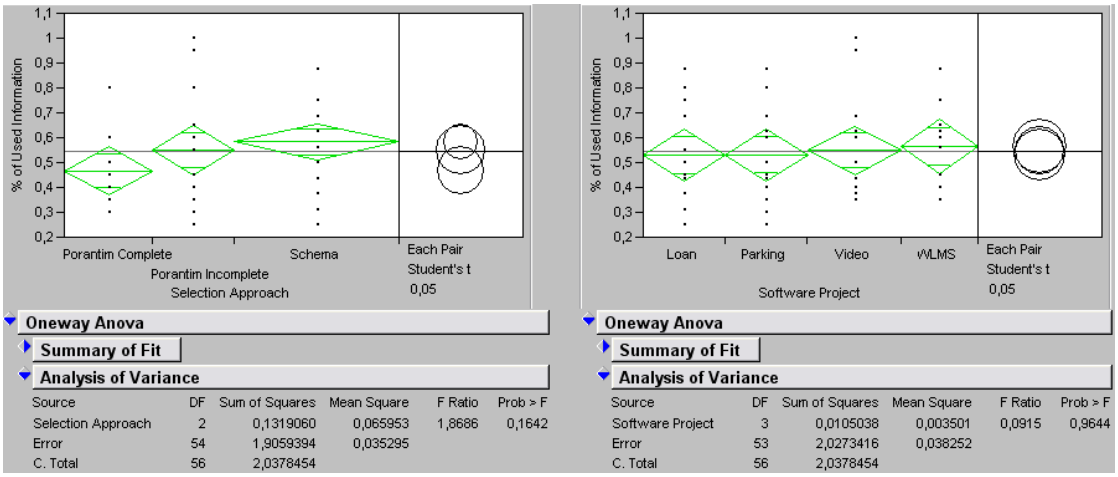


Figura 5.28. ANOVA para Completude (Percentual de Informações usadas)

Sendo assim, os resultados indicam que nenhum dos fatores investigados (abordagem de seleção e projeto de software) possui influência na redução do percentual de informações usadas para seleção de TTBMs em projetos de software.

5.5.2.2 Completude – Quantidade de Informações Faltando

- **Análise de *Outlier***

Aplicando-se análise de *outlier*, observa-se que um participante é avaliado como sendo um *outlier* na avaliação das respostas por abordagem de seleção e por projeto de software, pois relatou a ausência de 5 informações (Figura 5.29). Com isso, seus resultados foram removidos da análise desta variável (Figura 5.30).

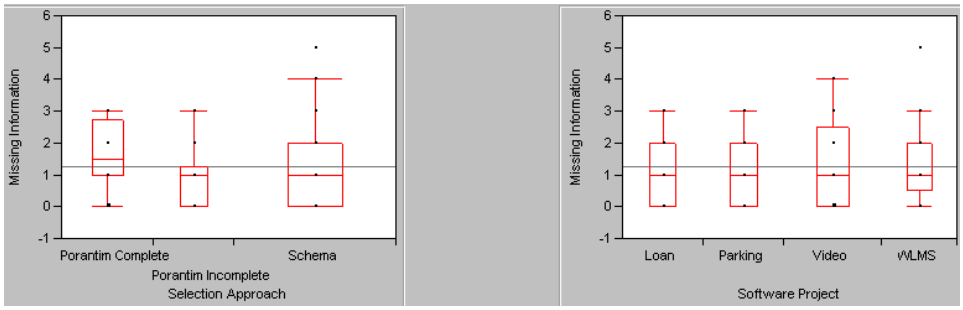


Figura 5.29. Análise de *Outlier* – Completude (Quantidade de Informações faltando) – Antes da Exclusão

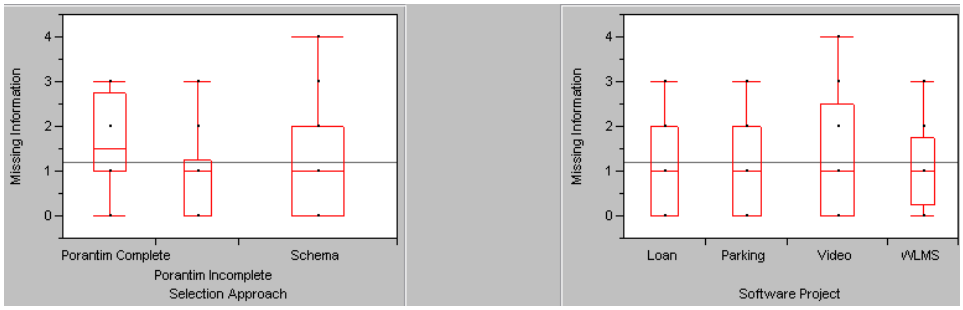


Figura 5.30. Análise de *Outlier* – Completude (Quantidade de Informações faltando) – Após a remoção do *Outlier*

- **Análise da Média**

Analisando a média entre as abordagens de seleção, observa-se que existe uma diferença significativa entre as médias obtidas pela versão completa de *Porantim* (1,56) em relação às outras duas abordagens de seleção (Esquema de Caracterização = 1,11 ; *Porantim* incompleta = 0,92). Analisando a média entre os projetos de software, percebemos que existe uma grande diferença de valores entre o projeto Vídeo (1,35) para com os demais projetos (1,08 ; 1,14 ; 1,14), o que indica que as características de um projeto poderiam influenciar de alguma forma no total de informações faltando para se tomar a decisão sobre a seleção de TTBMs. Tais resultados estão descritos na Tabela 5.24 e Figura 5.31.

Tabela 5.24. Média e Desvio Padrão – Completude (Quantidade de Informações faltando)

Fator	Alternativa	Média	Desvio Padrão
Abordagem de Seleção	EC	1,11	1,15
	PI	0,92	1,07
	PC	1,56	1,09
Projeto de Software	(V)	1,35	1,36
	(B)	1,14	1,16
	(E)	1,14	1,02
	(R)	1,08	0,90

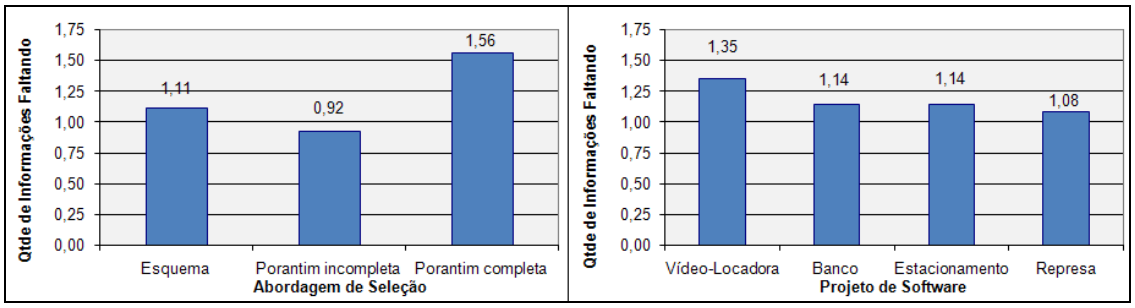


Figura 5.31. Média da Quantidade de Informações faltando por Abordagem de Seleção e Projeto de Software

- **Análise de Variância – ANOVA**

Aplicando-se testes estatísticos, percebe-se que, apesar do valor médio do total de informações faltando ser menor para a versão incompleta de *Porantim*, não existe diferença estatística entre as abordagens, como pode ser observado pela interseção entre todos os círculos representando cada abordagem de seleção e pelo *p-value* que ficou em 0,2712, superior ao valor 0,05 que representa o valor de significância estatística (taxa de erro) de 5% (ver lado esquerdo da Figura 5.32).

Fazendo a mesma análise para avaliar o impacto do projeto de software no total de informações faltando, percebe-se que não existe diferença estatística entre eles, como pode ser observado pela interseção entre todos os círculos representando cada projeto de software (eles estão quase sobrepostos) e pelo *p-value* que ficou em 0,9205, superior ao valor 0,05 que representa o valor de significância estatística (taxa de erro) de 5% (ver lado direito da Figura 5.32).

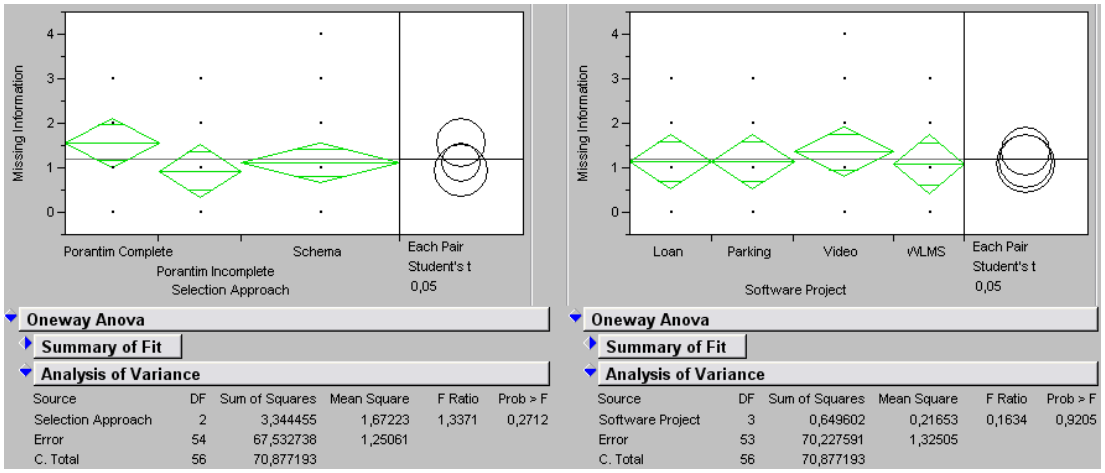


Figura 5.32. ANOVA para Completude (Quantidade de Informações faltando)

Sendo assim, os resultados indicam que nenhum dos fatores investigados (abordagem de seleção e projeto de software) possui influência na redução da quantidade de informações faltando para caracterização de TTBMs.

5.5.2.3 Efetividade – Percentual de Escolhas Corretas

- **Análise de *Outlier***

Aplicando-se análise de *outlier*, observa-se que nenhum participante é avaliado como sendo um *outlier* na avaliação das respostas por abordagem de seleção e por projeto de software (Figura 5.33).

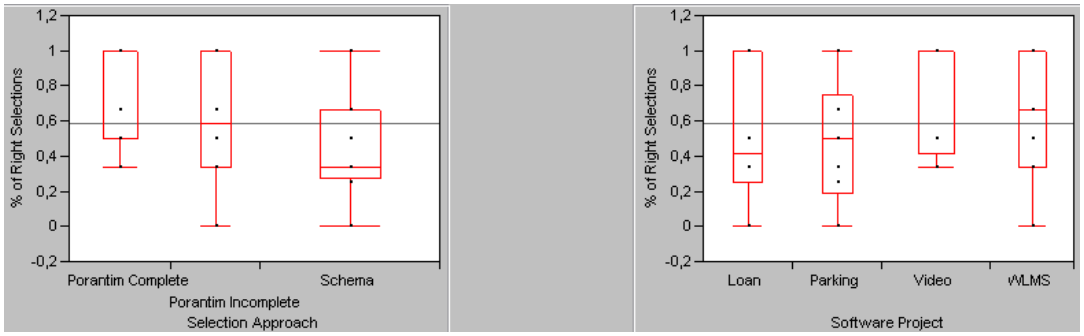


Figura 5.33. Análise de *Outlier* – Efetividade (Percentual de Escolhas Corretas)

- **Análise da Média**

Analisando a média entre as abordagens de seleção, observa-se que existe uma diferença significativa entre as médias obtidas pela versão completa de *Porantim* (79,16% de acerto) em comparação com sua versão incompleta (61,90%) e principalmente com o esquema de caracterização (45,53%). Isso indica que a versão completa de *Porantim* apoiaria no aumento do percentual de escolhas corretas de TTBM em projetos de software. Analisando a média entre os projetos de software, percebemos uma maior proximidade entre os projetos Empréstimo (48,80% de acerto) e Estacionamento (49,40%), e uma determinada distância para os projetos Controle de Represa (61,53%) e Vídeo (72,54%). Tais resultados estão descritos na Tabela 5.25 e Figura 5.34.

Tabela 5.25. Média e Desvio Padrão – Completude (% de Escolhas Corretas)

Fator	Alternativa	Média	Desvio Padrão
Abordagem de Seleção	EC	45,53%	34,87%
	PI	61,90%	33,60%
	PC	79,16%	25,45%
Projeto de Software	(V)	72,54%	30,58%
	(B)	48,80%	37,81%
	(E)	49,40%	36,17%
	(R)	61,53%	32,19%

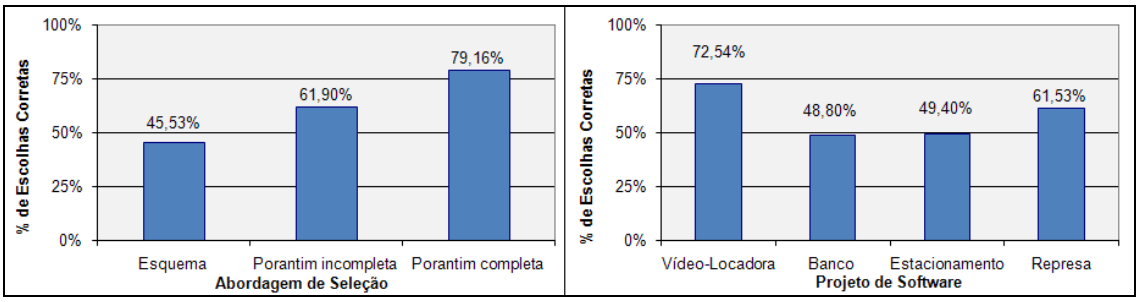


Figura 5.34. Média do Percentual de Escolhas Corretas por Abordagem de Seleção e Projeto de Software

- **Análise de Variância – ANOVA**

Aplicando-se testes estatísticos, percebe-se que existe diferença estatística entre as abordagens *Porantim* Completa e o Esquema de Caracterização, como pode ser observado pela não-interseção entre os círculos representando cada abordagem e pelo *p-value* que ficou em 0,0060, inferior ao valor 0,05 que representa o valor de significância estatística (taxa de erro) de 5% (ver lado esquerdo da Figura 5.35). Percebe-se ainda que a versão incompleta de *Porantim* não possui diferença estatística em relação às outras duas abordagens, o que significa que apenas incrementar o conjunto de atributos de caracterização de TTBMs apesar de reduzir o tempo gasto para seleção, não é suficiente para prover melhorias significativas nessa variável, o que só foi possível a partir da versão completa de *Porantim*, com indicadores e gráficos de apoio à seleção de TTBMs.

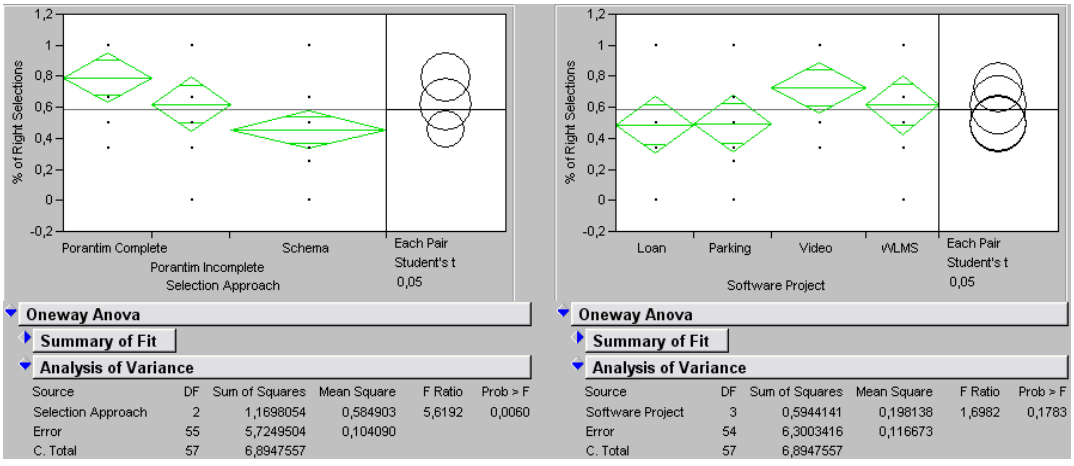


Figura 5.35. ANOVA para Efetividade (Percentual de Escolhas Corretas)

Fazendo a mesma análise para avaliar o impacto do projeto de software no percentual de seleções corretas, percebe-se que não existe diferença estatística entre eles, como pode ser observado pela interseção entre todos os círculos representando

cada projeto de software e pelo *p-value* que ficou em 0,1783, superior ao valor 0,05 que representa o valor de significância estatística (taxa de erro) de 5% (ver lado direito da Figura 5.35).

Sendo assim, os resultados indicam que o fator avaliado que influencia no aumento do percentual de seleções de TTBMs corretas em projetos de software seria a abordagem de seleção adotada.

5.5.2.4 Eficiência – Tempo de Resposta

- **Análise de *Outlier***

Aplicando-se análise de *outlier*, observa-se que existem 2 (dois) pontos relacionados à abordagem Esquema de Caracterização avaliados como *outliers*. Esses dois pontos são provenientes do mesmo participante, indicando que este participante realizou o estudo em uma quantidade de tempo significantemente superior aos demais, de forma que pode impactar na análise dos dados (Figura 5.36). Sendo assim, os dados deste participante foram removidos da análise dos dados (Figura 5.37).

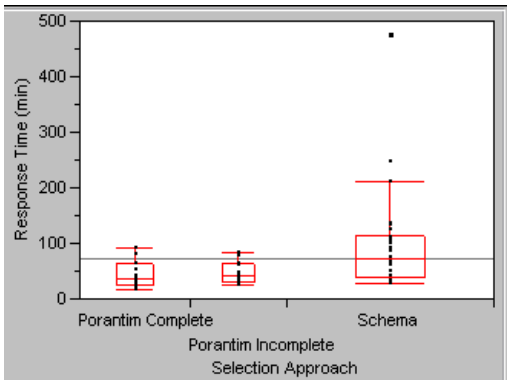


Figura 5.36. Análise de *Outlier* – Eficiência (Tempo de Resposta) – Antes da Exclusão

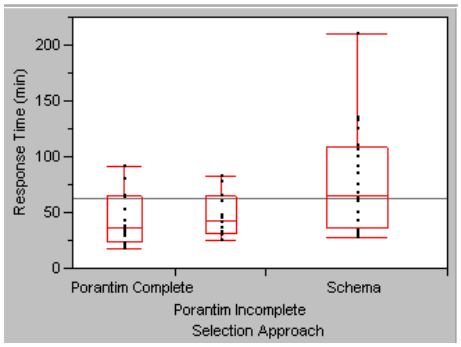


Figura 5.37. Análise de *Outlier* – Eficiência (Tempo de Resposta) – Após a exclusão

- **Análise da Média**

Analisando a média entre as abordagens de seleção, observa-se que existe diferença não muito significativa entre a média obtida pela versão completa de *Porantim* (44,06 min) em comparação com sua versão incompleta (47,71 min), mas existe uma diferença significativa em relação ao esquema de caracterização (79,25 min). Isso sugere que *Porantim*, independentemente de sua versão, apoiaria na redução do tempo gasto para a seleção de TTBMs em projetos de software. Analisando a média entre os projetos de software, percebemos uma diferença entre a média obtida para o projeto Estacionamento (51,50 min) em relação aos demais projetos (61,29 ; 67,53 ; 67,92 min). Este projeto possui um domínio simples em relação aos demais, o que pode justificar tal tempo. Tais resultados estão descritos na Tabela 5.26 e Figura 5.38.

Tabela 5.26. Média e Desvio Padrão – Eficiência (Tempo de Resposta)

Fator	Alternativa	Média	Desvio Padrão
Abordagem de Seleção	EC	79,25 min	50,36 min
	PI	47,71 min	18,97 min
	PC	44,06 min	22,85 min
Projeto de Software	(V)	61,29 min	28,55 min
	(B)	67,92 min	53,10 min
	(E)	51,00 min	29,05 min
	(R)	67,53 min	53,39 min

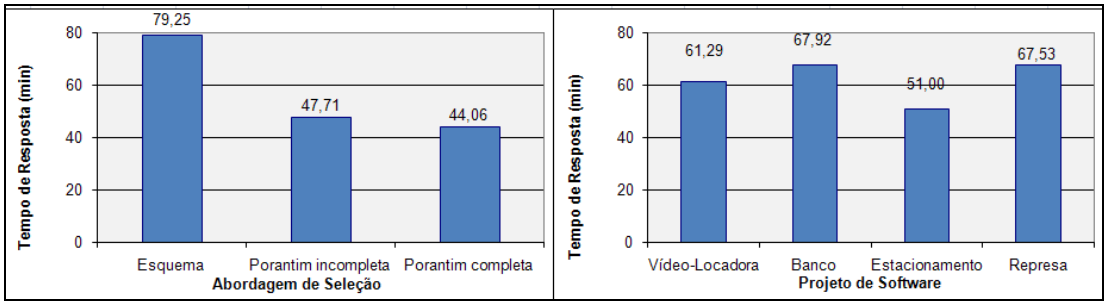


Figura 5.38. Média do Tempo de Resposta por Abordagem de Seleção e Projeto de Software

- **Análise de Variância – ANOVA**

Aplicando-se testes estatísticos, percebe-se que existe diferença estatística entre as abordagens *Porantim* Completa e o Esquema de Caracterização, como pode ser observado pela não-interseção entre os círculos representando cada abordagem e pelo *p-value* que ficou em 0,0064, inferior ao valor 0,05 que representa o valor de significância estatística (taxa de erro) de 5% (ver lado esquerdo da Figura 5.39). Percebe-se ainda que a versão incompleta de *Porantim* não possui diferença estatística em relação às outras duas abordagens (apesar de quase não haver interseção entre

os círculos representando as versões completa e incompleta de *Porantim*), o que significa que apenas incrementar o conjunto de atributos de caracterização de TTBMs apesar de reduzir o tempo gasto para seleção, não é suficiente para prover melhorias significativas nessa variável, o que só foi possível a partir da versão completa de *Porantim*, com indicadores e gráficos de apoio à seleção de TTBMs.

Fazendo a mesma análise para avaliar o impacto do projeto de software no tempo gasto para seleção, percebe-se que não existe diferença estatística entre eles, como pode ser observado pela interseção entre todos os círculos representando cada projeto de software e pelo *p-value* que ficou em 0,7090, superior ao valor 0,05 que representa o valor de significância estatística (taxa de erro) de 5% (lado direito da Figura 5.39).

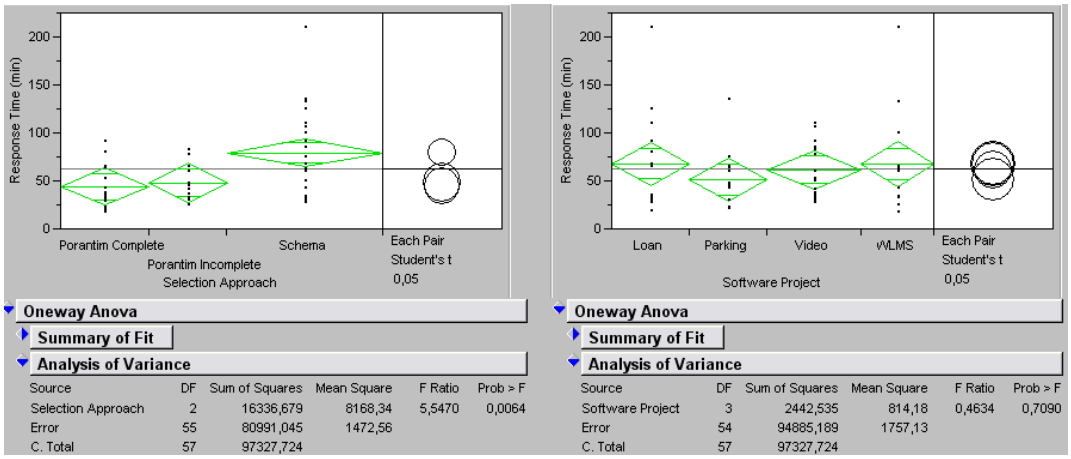


Figura 5.39. ANOVA para Eficiência (Tempo de Resposta)

Sendo assim, os resultados indicam que o fator avaliado que influencia na redução do tempo gasto para seleção de TTBMs em projetos de software seria a abordagem de seleção adotada.

5.5.2.5 Eficiência – Percentual de Escolhas Corretas / Percentual de Informações Usadas

A métrica percentual de escolhas corretas por percentual de informações usadas indica qual o esforço investido por cada abordagem de seleção para se atingir uma seleção mais precisa para um projeto de software. Quanto mais informações usadas para se obter escolhas corretas, menos eficiente seria a abordagem de seleção. Ou seja, quanto maior for este indicador, mais eficiente será a abordagem de seleção.

- **Análise de *Outlier***

Aplicando-se análise de *outlier*, observa-se que nenhum participante é avaliado como sendo um *outlier* na avaliação das respostas por abordagem de seleção e por projeto de software (Figura 5.40).

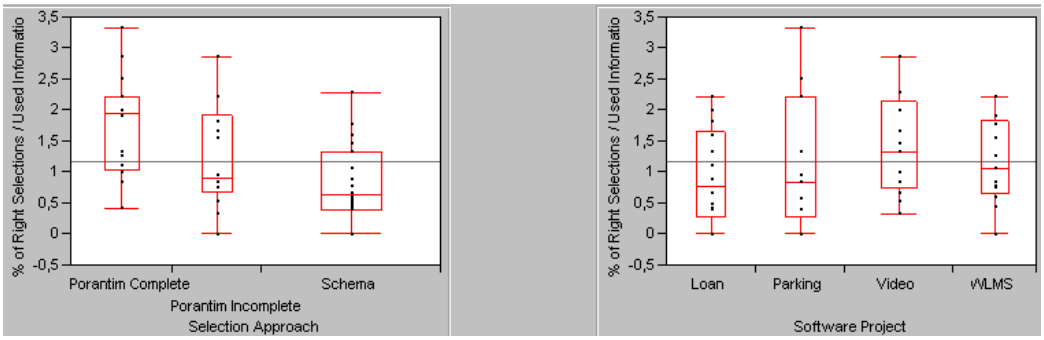


Figura 5.40. Análise de *Outlier* – Eficiência (% de Escolhas Corretas / % de Informações usadas)

- **Análise da Média**

Analisando a média entre as abordagens de seleção, observa-se que existe uma diferença significativa entre as médias obtidas pela versão completa de *Porantim* (1,76) em comparação com sua versão incompleta (1,24) e principalmente com o esquema de caracterização (0,81). Isso indica que a versão completa de *Porantim* possibilita obter um percentual maior de acertos na seleção de TTBMs usando uma quantidade menor de informações/atributos. Analisando a média entre os projetos de software, percebemos uma maior proximidade entre elas (0,92; 1,11; 1,18; 1,43), o que indica que o comportamento seria de certa forma similar entre os projetos. Tais resultados estão descritos na Tabela 5.27 e Figura 5.41.

Tabela 5.27. Média e Desvio Padrão – Eficiência (% de Escolhas Corretas / % de Informações usadas)

Fator	Alternativa	Média	Desvio Padrão
Abordagem de Seleção	EC	0,81	0,68
	PI	1,24	0,82
	PC	1,76	0,80
Projeto de Software	(V)	1,43	0,79
	(B)	0,92	0,76
	(E)	1,11	1,05
	(R)	1,18	0,70

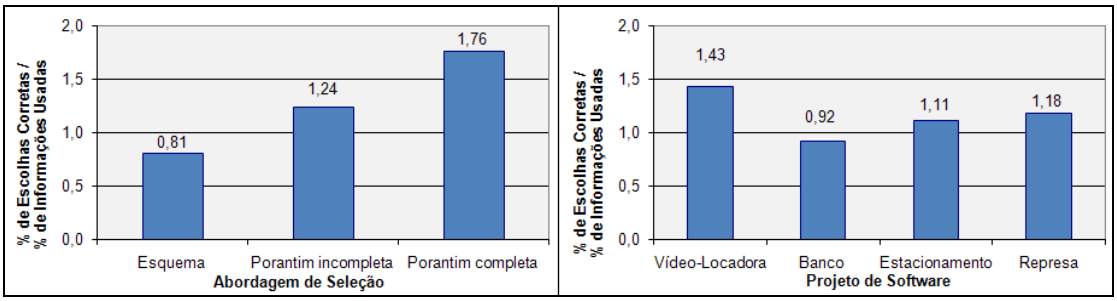


Figura 5.41. Média do "% de Escolhas Corretas / % de Informações usadas" por Abordagem de Seleção e Projeto de Software

- **Análise de Variância – ANOVA**

Aplicando-se testes estatísticos, percebe-se que existe diferença estatística entre as abordagens *Porantim* Completa e o Esquema de Caracterização, como pode ser observado pela não-interseção entre os círculos representando cada abordagem e pelo *p-value* que ficou em 0,0007, quase nulo e bem abaixo do valor 0,05 que representa o valor de significância estatística (taxa de erro) de 5% (ver lado esquerdo da Figura 5.42). Percebe-se ainda uma diferença significativa entre a versão incompleta de *Porantim* com sua versão completa, apesar de ainda haver uma interseção entre seus círculos, o que indica que as funcionalidades adicionais providas pela versão completa de *Porantim* contribuem significantemente para tornar a seleção de TTBMs mais eficiente e com o uso de uma quantidade menor de informações.

Fazendo a mesma análise para avaliar o impacto do projeto de software nesta variável, percebe-se que não existe diferença estatística entre eles, como pode ser observado pela interseção entre todos os círculos representando cada projeto de software e pelo *p-value* que ficou em 0,4049, superior ao valor 0,05 que representa o valor de significância estatística (taxa de erro) de 5% (ver lado direito da Figura 5.42).

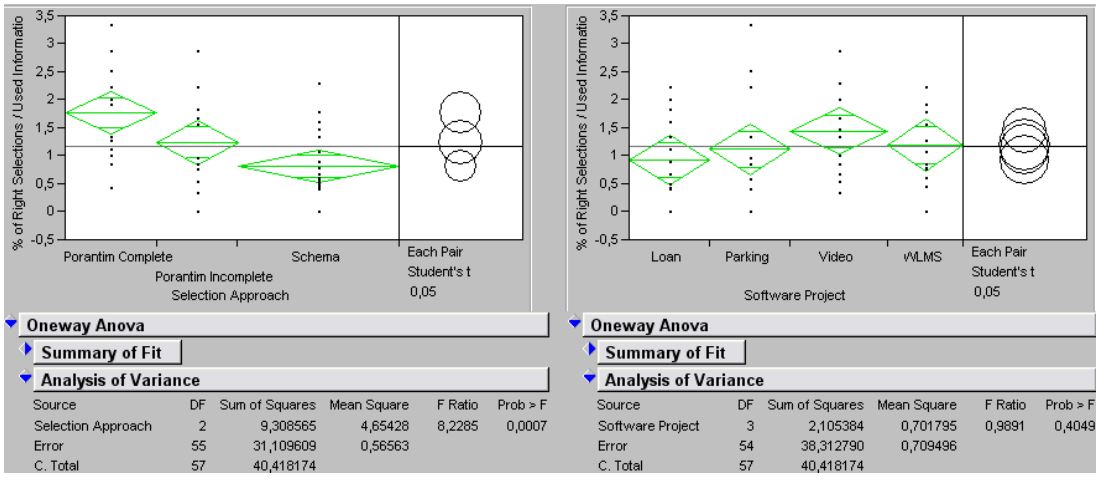


Figura 5.42. ANOVA para Eficiência (% de Escolhas Corretas / % de Informações Usadas)

Sendo assim, os resultados indicam que o fator avaliado que influencia no aumento do percentual de seleções de TTBMs corretas em projetos de software por informações usadas seria a abordagem de seleção adotada.

5.5.2.6 Eficiência – Percentual de Escolhas Corretas / Tempo de Resposta

A métrica percentual de escolhas corretas por tempo de resposta tenta expressar o dinamismo da abordagem de apoio à seleção de TTBMs em indicar um maior percentual de técnicas corretas para um projeto em uma menor quantidade de tempo (em minutos). Sendo assim, quanto maior for este valor, mais eficiente será a abordagem de seleção.

- **Análise de *Outlier***

Aplicando-se análise de *outlier*, observa-se que existem 4 (quatro) pontos de *outliers*: 3 na avaliação das respostas por abordagem de seleção e um outro na avaliação por projeto de software (Figura 5.43). Com isso, os dados de tais participantes foram removidos da análise desta variável (Figura 5.44).

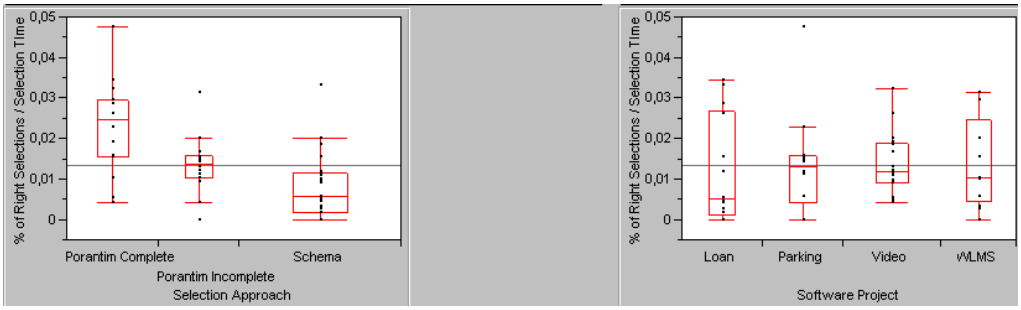


Figura 5.43. Análise de *Outlier* – Eficiência (% de Escolhas Corretas / Tempo de Resposta) – Antes da Exclusão

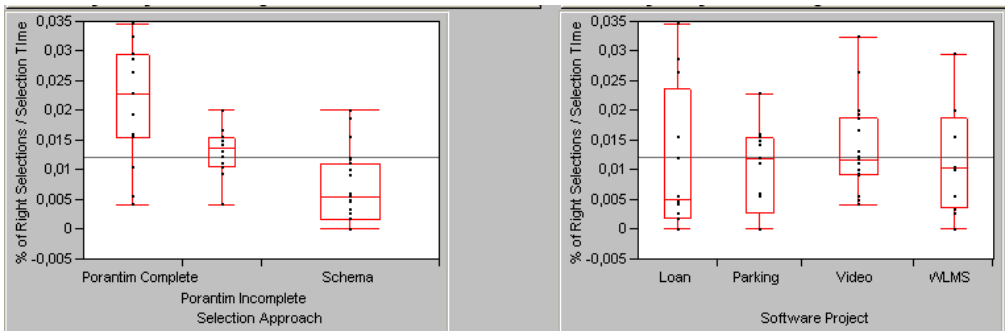


Figura 5.44. Análise de *Outlier* – Eficiência (% de Escolhas Corretas / Tempo de Resposta) – Após a remoção dos *outliers*

- **Análise da Média**

Analisando a média entre as abordagens de seleção, observa-se que existe uma grande diferença entre as médias obtidas pela versão completa de *Porantim* (2,10% acertos por minuto) em comparação com sua versão incompleta (1,30% acertos por minuto) e principalmente com o esquema de caracterização (0,67% acertos por minuto). Isso indica que a versão completa de *Porantim* possibilita obter um percentual maior percentual de acertos na seleção de TTBMs em uma quantidade mais reduzida de tempo. Analisando a média entre os projetos de software, percebemos que os valores são bastante próximos entre todos os projetos (1,02%; 1,12%; 1,26%; 1,37%). Tais resultados estão descritos na Tabela 5.28 e Figura 5.45.

Tabela 5.28. Média e Desvio Padrão – [% de Escolhas Corretas / Tempo de Resposta]

Fator	Alternativa	Média	Desvio Padrão
Abordagem de Seleção	EC	0,67 %/min	0,58 %/min
	PI	1,30 %/min	0,40 %/min
	PC	2,10 %/min	0,96 %/min
Projeto de Software	(V)	1,37 %/min	0,76 %/min
	(B)	1,12 %/min	1,21 %/min
	(E)	1,02 %/min	0,72 %/min
	(R)	1,26 %/min	0,97 %/min

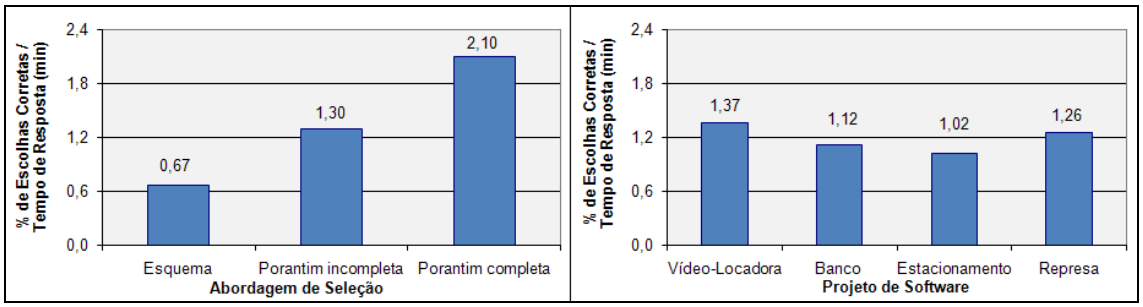


Figura 5.45. Média do "% de Escolhas Corretas / Tempo de Resposta" por Abordagem de Seleção e Projeto de Software

- **Análise de Variância – ANOVA**

Aplicando-se testes estatísticos, percebe-se que existe diferença estatística entre as abordagens *Porantim* Completa em relação às outras duas abordagens avaliadas, como pode ser observado pela não-interseção entre o círculo representando a abordagem *Porantim* completa e o círculo representando o Esquema de Caracterização, além do *p-value* que ficou abaixo de 0,0001, quase nulo (ver lado esquerdo da Figura 5.46). Percebe-se ainda uma diferença estatística entre a versão incompleta de *Porantim* em relação ao esquema de caracterização, porém seus círculos não possuem interseção. Este resultado indica que os atributos específicos

para caracterização de TTBMs inseridos por *Porantim* contribuem significantemente para tornar a seleção de TTBMs mais eficiente em relação ao percentual de acertos por unidade de tempo (minutos), porém as funcionalidades adicionais providas pela versão completa de *Porantim* tornam essa eficiência ainda maior.

Fazendo a mesma análise para avaliar o impacto do projeto de software nesta variável, percebe-se que não existe uma diferença estatística entre eles, como pode ser observado pela interseção entre todos os círculos representando cada projeto de software (na verdade quase todos os círculos estão sobrepostos) e pelo *p-value* que ficou em 0,7427, superior ao valor 0,05 que representa o valor de significância estatística (taxa de erro) de 5% (ver lado direito da Figura 5.46).

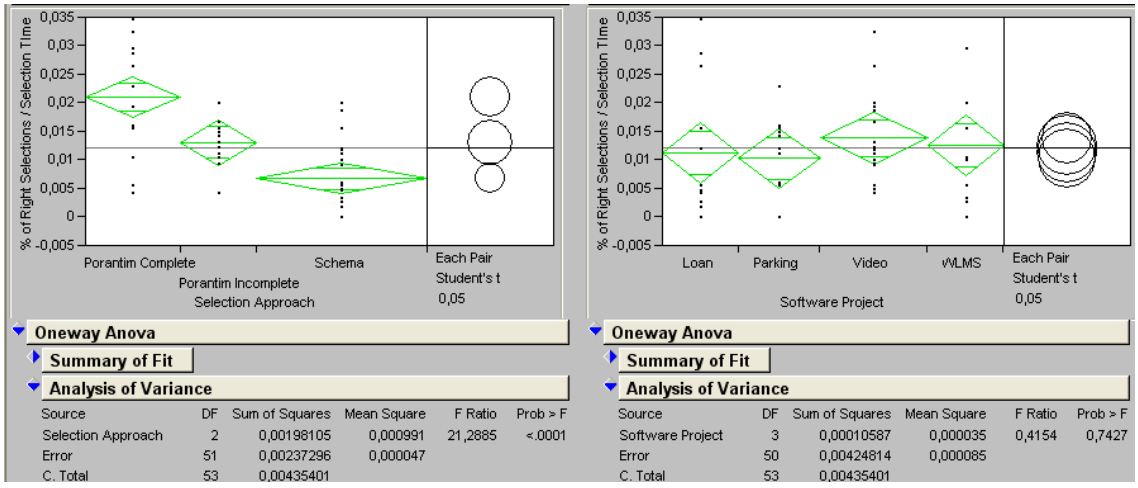


Figura 5.46. ANOVA para Eficiência (% de Escolhas Corretas / Tempo de Resposta)

Sendo assim, os resultados indicam que o fator avaliado que influencia no aumento do percentual de seleções de TTBMs corretas em projetos de software por unidade de tempo seria a abordagem de seleção adotada.

5.5.2.7 Usabilidade – Quantidade de Problemas relatados durante a seleção

- **Análise de *Outlier***

Aplicando-se análise de *outlier*, observa-se que nenhum participante é avaliado como sendo um *outlier* na avaliação das respostas por abordagem de seleção e por projeto de software (Figura 5.47).

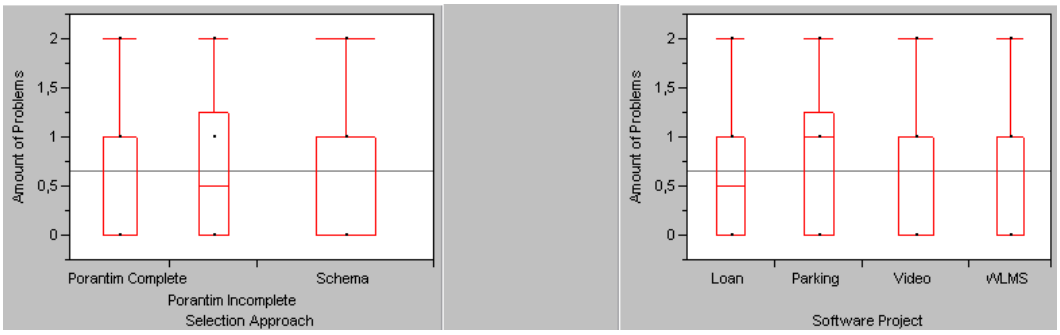


Figura 5.47. Análise de *Outlier* – Usabilidade (Quantidade de Problemas)

- **Análise da Média**

Analisando a média entre as abordagens de seleção, observa-se que existe uma diferença significativa entre as médias obtidas pelas versões completa e incompleta de *Porantim* (0,81 e 0,71) em comparação com o esquema de caracterização (0,53). Isso indica que os participantes que usaram *Porantim* relataram mais problemas ou dificuldades em selecionar TTBMs. Analisando a média entre os projetos de software, percebemos uma maior proximidade entre elas (0,58; 0,61; 0,64; 0,78), o que indica que as caractersticas do projeto não seriam um fator que influenciasse na quantidade de problemas relatados durante a seleção de TTBMs. Tais resultados estão descritos na Tabela 5.29 e Figura 5.48.

Tabela 5.29. Média e Desvio Padrão – Usabilidade (Quantidade de Problemas)

Fator	Alternativa	Média	Desvio Padrão
Abordagem de Seleção	EC	0,53	0,69
	PI	0,71	0,82
	PC	0,81	0,75
Projeto de Software	(V)	0,58	0,79
	(B)	0,64	0,74
	(E)	0,78	0,80
	(R)	0,61	0,65

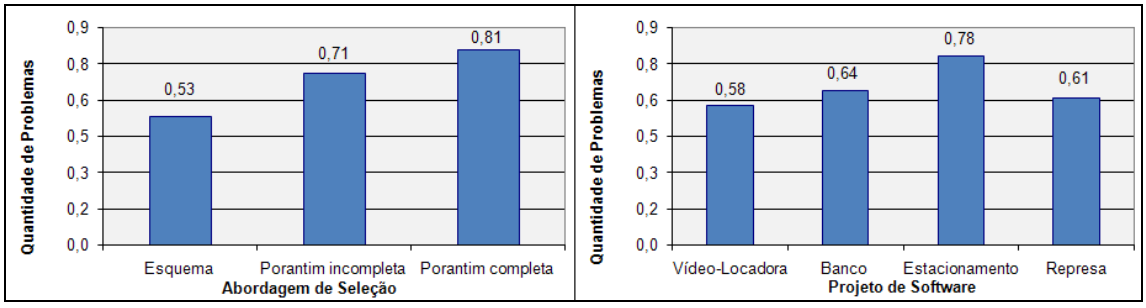


Figura 5.48. Média da Quantidade de Problemas por Abordagem de Seleção e Projeto de Software

- **Análise de Variância – ANOVA**

Aplicando-se testes estatísticos, percebe-se que apesar dos valores de média obtidos por *Porantim* serem superiores, não existe diferença estatística entre as abordagens de seleção, como pode ser observado pela interseção entre os círculos representando cada abordagem e pelo *p-value* que ficou em 0,4691, acima do valor 0,05 que representa o valor de significância estatística (taxa de erro) de 5% (ver lado esquerdo da Figura 5.49). Percebe-se ainda que os valores obtidos entre as versões incompletas e completas de *Porantim* são quase os mesmos, e os círculos que descrevem cada abordagem estão quase sobrepostos.

Fazendo a mesma análise para avaliar o impacto do projeto de software nesta variável, percebe-se que não existe diferença estatística entre eles. Isso pode ser observado pela interseção entre todos os círculos representando cada projeto de software e pelo *p-value* que ficou em 0,8977, superior ao valor 0,05 que representa o valor de significância estatística (taxa de erro) de 5% (ver lado direito da Figura 5.49).

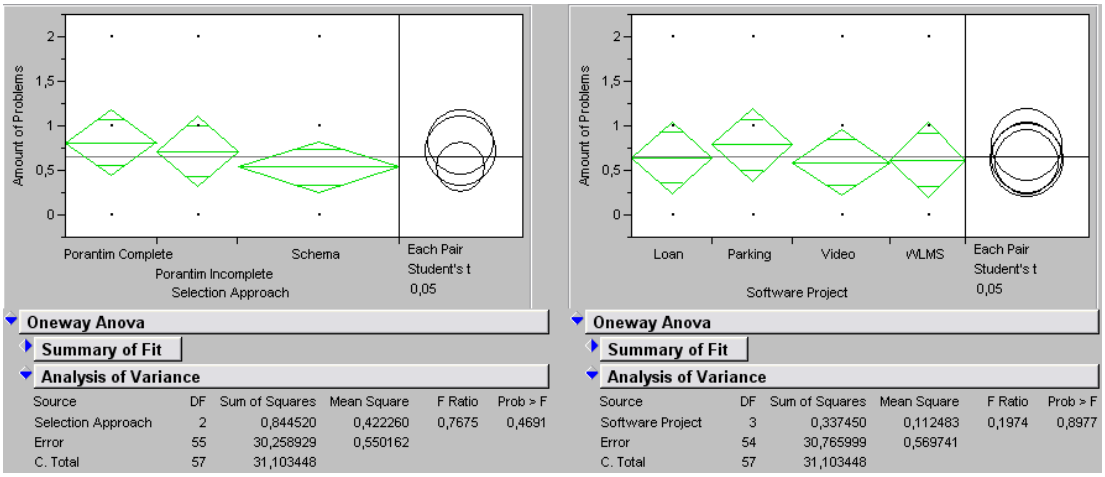


Figura 5.49. ANOVA para Usabilidade (Quantidade de Problemas)

Sendo assim, os resultados indicam que nenhum dos fatores investigados (abordagem de seleção e projeto de software) possui influência na redução ou aumento da quantidade de problemas relatados durante a seleção de TTBMs.

5.5.3 Resumo dos Resultados Quantitativos

Esta Seção apresenta um resumo dos resultados obtidos ao longo da análise quantitativa dos dados deste estudo para as duas instâncias executadas: POS (Tabela 5.30) e GRAD (Tabela 5.31). Como podem ser observados, os resultados estatísticos (indicação de qual fator influencia no processo de seleção de TTBMs) obtidos foram

exatamente os mesmos para todas as variáveis analisadas neste estudo considerando as duas instâncias executadas.

Os resultados sugerem que o fator abordagem de seleção possui influência direta na *efetividade* e *eficiência* do processo de seleção de TTBMs. Segundo as análises realizadas, a versão completa de *Porantim* provê melhorias para a *efetividade* (percentual de escolhas corretas) e *eficiência* (tempo de resposta; percentual de escolhas corretas por percentual de informações usadas; percentual de escolhas corretas por tempo de resposta) para o processo de seleção de TTBMs. Por outro lado, os aspectos *completude* e *usabilidade* não são influenciados pela abordagem de seleção.

Ao analisar os dados por projetos de software, os dados obtidos nas duas instâncias indicam que o projeto de software não é um fator que influencie no processo de seleção de TTBMs.

Tabela 5.30. Resumo dos Resultados – Instância POS

Categoria	Variável	Outlier?	Média	*P-value*	Quem influencia?
Completude	Percentual de Informações Usadas	NÃO	EC: 68,87% PI: 60,00% PC: 52,50%	0,0608	NENHUMA
		NÃO	V: 55,50% B: 65,90% E: 59,12% R: 63,29%	0,5254	NENHUM
	Quantidade de Informações Faltando	SIM	EC: 1,42 PI: 1,00 PC: 1,66	0,3664	NENHUMA
		SIM	V: 1,60 B: 1,45 E: 1,00 R: 1,50	0,6348	NENHUM
Efetividade	Percentual de Escolhas Corretas	NÃO	EC: 50,00% PI: 68,33% PC: 86,11%	0,0215	*PORANTIM* COMPLETA
		NÃO	V: 73,33% B: 57,57% E: 68,33% R: 60,60%	0,7682	NENHUM
Eficiência	Tempo de Resposta (minutos)	NÃO	EC: 65,50 min PI: 58,10 min PC: 45,25 min	0,0141	*PORANTIM* COMPLETA
		NÃO	V: 54,30 min B: 53,64 min E: 61,91 min R: 62,00 min	0,6361	NENHUM
	Percentual de Escolhas Corretas / Percentual de Informações Usadas	NÃO	EC: 0,73 PI: 1,38 PC: 1,71	0,0019	*PORANTIM* COMPLETA
		NÃO	V: 1,38 B: 0,94 E: 1,24 R: 1,14	0,6799	NENHUM
	Percentual de Escolhas Corretas / Tempo de Resposta	SIM	EC: 0,78 (%/min) PI: 1,28 (%/min) PC: 1,89 (%/min)	< 0,0001	*PORANTIM* COMPLETA
		SIM	V: 1,51 (%/min) B: 1,21 (%/min) E: 1,10 (%/min) R: 0,93 (%/min)	0,5921	NENHUM
Usabilidade	Quantidade de Problemas relatados durante a seleção de técnicas de TBM	SIM	EC: 1,05 PI: 0,55 PC: 0,72	0,2074	NENHUMA
		SIM	V: 1,10 B: 0,67 E: 0,55 R: 1,00	0,3424	NENHUM

Tabela 5.31. Resumo dos Resultados – Instância GRAD

Categoria	Variável	Outlier?	Média	P-value	Quem influencia?
Completude	% de Informações Usadas	SIM	EC: 58,25% PI: 55,00% PC: 46,66%	0,1642	NENHUMA
		NÃO	V: 54,84% B: 53,12% E: 53,21% R: 56,53%	0,9644	NENHUM
	Quantidade de Informações Faltando	SIM	EC: 1,11 PI: 0,92 PC: 1,56	0,2712	NENHUMA
		SIM	V: 1,35 B: 1,14 E: 1,14 R: 1,08	0,9205	NENHUM
Efetividade	% de Escolhas Corretas	NÃO	EC: 45,53% PI: 61,90% PC: 79,16%	0,0060	*PORANTIM* COMPLETA
		NÃO	V: 72,54% B: 48,80% E: 49,40% R: 61,53%	0,1783	NENHUM
Eficiência	Tempo de Resposta (minutos)	SIM	EC: 79,25 min PI: 47,71 min PC: 44,06 min	0,0064	*PORANTIM* COMPLETA
		NÃO	V: 61,29 min B: 67,92 min E: 51,50 min R: 67,53 min	0,7090	NENHUM
	% de Escolhas Corretas / % de Informações Usadas	NÃO	EC: 0,81 PI: 1,24 PC: 1,76	0,0007	*PORANTIM* COMPLETA
		NÃO	V: 1,43 B: 0,92 E: 1,11 R: 1,18	0,4049	NENHUM
	% de Escolhas Corretas / Tempo de Resposta	SIM	EC: 0,67 (%/min) PI: 1,30 (%/min) PC: 2,10 (%/min)	< 0,0001	*PORANTIM* COMPLETA
		SIM	V: 1,37 (%/min) B: 1,12 (%/min) E: 1,02 (%/min) R: 1,26 (%/min)	0, 7427	NENHUM
Usabilidade	Quantidade de Problemas relatados durante a seleção de técnicas de TBM	NÃO	EC: 0,53 PI: 0,71 PC: 0,81	0,4691	NENHUMA
		NÃO	V: 0,58 B: 0,64 E: 0,78 R: 0,61	0,8977	NENHUM

5.6 Análise Qualitativa dos Resultados

A análise qualitativa dos dados foi realizada para cada aspecto avaliado neste estudo, incluindo o aspecto “Satisfação do Usuário”, analisando:

- Os dados quantitativos apresentados na Seção 5.5 e o desempenho dos participantes ao longo das duas seleções que cada um executou;
- Os dados quantitativos entre as diferentes instâncias executadas (POS e GRAD);
- Extraindo informações a partir dos questionários F2 e F9 (apresentados no APÊNDICE B – Seção AP B.1).

5.6.1 Análise Qualitativa do Aspecto Completude

- **Análise das Informações Usadas**

O fato de a versão completa de *Porantim* usar apenas um subconjunto dos atributos de caracterização para o cálculo do grau de adequação entre TTBMs e projeto de software pode ter direcionado os participantes apenas no uso desses atributos no processo de seleção de TTBMs.

Analisando quais atributos são os mais usados, entre os 11 atributos de caracterização mais usados pelos participantes usando *Porantim* nas instâncias POS e GRAD, 9 são usados para o cálculo do grau de adequação, o que indica que esta funcionalidade de *Porantim* ajuda a filtrar os atributos relevantes para a seleção de TTBMs (Tabela 5.32).

Tabela 5.32. Atributos de Caracterização mais usados no processo de seleção

Atributos Usados	POS		GRAD	
	Esquema	*Porantim*	Esquema	*Porantim*
Nível(is) de Teste	85,00%	95,45%	96,67%	93,33%
Características de qualidade do software	90,00%	90,91%	90,00%	93,33%
Tipo de Técnica de Teste	-	90,91%	-	90,00%
Tipos de Falhas que permite revelar	95,00%	90,91%	100,00%	90,00%
Tecnologia usada para modelagem	-	81,82%	76,67%	73,33%
Linguagem de programação	95,00%	81,82%	73,33%	60,00%
Modelo comportamental/estrutural	-	77,27%	-	56,67%
Paradigma de desenvolvimento de software	85,00%	68,18%	-	56,67%
Entradas requeridas	-	68,18%	-	56,67%
Plataforma de execução do software	80,00%	59,09%	70,00%	53,33%
Custo associado à ferramenta	95,00%	54,55%	60,00%	50,00%
Complexidade dos passos não-automatizados	50,00%	54,55%	70,00%	46,67%
Habilidade requerida para ser operada	85,00%	45,45%	56,67%	46,67%
Ferramenta de Apoio	45,00%	36,36%	60,00%	43,33%
Nível de Automação	60,00%	36,36%	-	26,67%
Existência de um mecanismo de rastreabilidade	-	27,27%	26,67%	26,67%
Existência de um verificador de modelo	-	22,73%	23,33%	26,67%
Resultados gerados	25,00%	18,18%	30,00%	20,00%
Plataforma da ferramenta de apoio	40,00%	13,64%	-	16,67%
Critério de Geração dos Testes	20,00%	4,55%	26,67%	16,67%
Compreensibilidade	80,00%	-	60,00%	-
Dependências	35,00%	-	13,33%	-

Analisando os novos atributos introduzidos por *Porantim* que são específicos de TBM, 4 dos 6 novos atributos estão listados entre os 10 mais utilizados para a seleção de TTBMs, o que indica que realmente eles possuem influência na seleção de TTBMs.

Além disso, os atributos do esquema que foram removidos em *Porantim* em relação ao Esquema de Caracterização foram praticamente os menos usados pelos participantes que aplicaram o Esquema de Caracterização, o que indica que eles não possuem tanta influência no processo de seleção de TTBM.

Percebe-se ainda que alguns atributos são muito utilizados com o esquema de caracterização por não existirem atributos específicos de caracterização de TTBMs, de forma que eles tentem suprir a ausência de atributos mais relevantes (ex: atributos complexidade dos passos não-automatizados e nível de automação).

- **Análise das Informações Faltando**

Ao total, 16 informações faltando foram relatadas na instância POS e 9 informações na instância GRAD. Ao total, 17 itens foram relatados após descartar as sobreposições (Tabela 5.33). As informações comuns a ambas as abordagens estão relacionadas à necessidade de melhor detalhamento ou padronização de alguns campos e uma estimativa de esforço para usar uma TTBM, algo que é provido pelo processo de seleção de TTBMs que compõe a abordagem *Porantim*, mas não foi utilizado neste estudo. As informações faltando no Esquema de Caracterização estão relacionadas à ausência de atributos de caracterização de TTBMs, tais como o modelo ou tecnologia adotada pela técnica, o que está incluído em *Porantim*.

Tabela 5.33. Informações faltando relatadas pelos participantes do estudo

Informações Faltando	POS		GRAD	
	Esquema	*Porantim*	Esquema	*Porantim*
Detalhamento do Custo da Ferramenta	7	6	7	18
Descrição da Plataforma da Ferramenta	5	4	4	7
Tamanho de aplicação que as técnicas atendem	3	4	-	-
Mudança nos Requisitos (apoio à rastreabilidade)	2	3	3	-
Tipo de técnica de teste contemplada pela TTBM	5	-	7	-
Indicação da adequação da equipe às abordagens	-	4	1	-
Detalhamento dos critérios de geração dos testes	-	3	-	-
Modelo adotado pela TTBM	2	-	5	-
Indicação da dificuldade de aprendizado da abordagem	-	2	-	-
Grau de dificuldade de aprendizado da linguagem	1	1	-	1
Estimativa de tempo para uso da ATBM	1	1	2	3
Indicação de qual distante são as características das técnicas de TBM	-	1	-	-
Detalhamento da Linguagem de Programação	-	1	-	-
Detalhamento da compreensibilidade	1	-	-	-
Atributo indicando um apoio à verificação do modelo	1	-	-	-
Informações sobre os resultados/saídas gerados pela técnica de TBM	-	-	1	-

Analisando quais atributos que estariam faltando, a maioria estaria relacionada ao detalhamento do custo e plataforma que apóia a técnica, porém não é propósito das abordagens registrar um valor monetário associado à aquisição da ferramenta. Apenas um indicador se esta é gratuita ou não é definido.

Além disso, o quarto item mais citado indica a necessidade de apoiar a mudança dos requisitos no software (pois esse atributo existe na caracterização do projeto de software), mas esse atributo está presente em *Porantim* através da descrição da existência de um mecanismo que apóia a rastreabilidade entre os testes e o software.

Dos 5 atributos relatados como ausentes no Esquema de Caracterização nas duas instâncias, 3 são providos por *Porantim*. São eles:

- Tipo de técnica de teste contemplada pela TTBM.
- Modelo adotado pela TTBM.
- Atributo indicando um apoio à verificação do modelo.
- Mudança dos requisitos (apoio à rastreabilidade).
- Informações sobre os resultados/saídas gerados pela TTBM.

Dos 4 atributos relatados como ausente em *Porantim*, 3 são providos ao longo do processo de apoio à seleção de TTBMs que compõe *Porantim*, mas que não foram usados no estudo por tratarmos apenas a etapa de seleção individual de TTBM. São eles:

- Indicação da adequação da equipe às abordagens: *Porantim* provê uma análise indicando o grau de adequação dos membros da equipe de teste selecionados para o projeto de software em relação às habilidades requeridas para usar uma TTBM.
- Indicação da dificuldade de aprendizado da abordagem: *Porantim* provê atributos indicando as habilidades requeridas para usar uma TTBM.
- Indicação de quão distante são as características das TTBMs: *Porantim* provê um indicador numérico descrevendo a distância conceitual entre as TTBMs e o projeto em questão.

5.6.2 Análise Qualitativa do Aspecto Efetividade

Pode ser observado que o percentual de seleções corretas não depende do fato de ser a primeira ou segunda seleção realizada pelo participante.

Além disso, na instância GRAD, o percentual de seleções corretas em cada abordagem foi inferior ao obtido com a instância POS:

- *Porantim* completa: 79% GRAD ; 86% POS;
- *Porantim* incompleta: 61% GRAD ; 68% POS;
- Esquema de Caracterização: 45% GRAD ; 50% POS.

Essa diferença pode refletir de certa forma a diferença do grau de experiência/conhecimento das duas amostras, pois os participantes da instância POS mesmo não tendo experiência na seleção de TTBMs, possuem mais experiência no contexto geral de Engenharia de Software. Com isso, acredita-se que o grau de experiência dos participantes afetaria no percentual de seleções corretas obtido por cada abordagem, independentemente de qual seja.

5.6.3 Análise Qualitativa do Aspecto Eficiência

- **Análise do Tempo de Resposta por Abordagem de Seleção**

Primeiramente, observa-se que o tempo médio de resposta obtido para a versão completa de *Porantim* na instância GRAD é quase o mesmo obtido na instância POS (44,06 GRAD ; 45,25 POS), sugerindo que o grau de experiência/conhecimento das pessoas não influencia muito no uso desta abordagem de seleção. Por outro lado, o tempo médio de resposta obtido para a versão incompleta de *Porantim* na instância GRAD foi bem menor que o obtido em POS (47,71 GRAD ; 58,10 POS), o que pode indicar que o fato de os alunos da graduação não terem participado de projetos reais, eles não possuem noção exata do impacto de más decisões a respeito da escolha de uma técnica em um projeto, de forma que eles executaram o exercício considerando o seu próprio critério de seleção.

O tempo médio obtido para o esquema de caracterização foi bem maior na instância GRAD que o obtido com a POS (79,25 GRAD ; 65,50 POS), o que pode indicar que apenas o uso do esquema de caracterização, sem os atributos de caracterização específicos da área de TBM, torna o processo de seleção de TTBMs mais difícil e custoso de ser realizado por pessoas menos experientes em realizar tal tarefa.

Analisando o percentual de escolhas corretas por unidade de tempo, se forem comparados os valores das médias obtidas nas duas instâncias, o valor obtido para a versão completa de *Porantim* foi maior na instância GRAD que o obtido com a instância POS (2,10% GRAD ; 1,89% POS), o que confirmaria que *Porantim* apresenta um efeito positivo com aqueles que possuem menos experiência em realizar tal tarefa. Os valores das médias para as outras duas abordagens também foram superiores na instância GRAD que os valores obtidos na instância POS, indicando que os

participantes da instância GRAD foram mais eficientes, em geral, que os participantes da instância POS.

- **Análise do Efeito de Aprendizagem durante a seleção**

Na instância POS, dentre os 21 participantes que finalizaram o estudo, 17 realizaram a segunda seleção com tempo inferior à primeira. A distância entre o tempo gasto na primeira seleção e na segunda chega a ser de 46 minutos. Já em relação aos 4 participantes que demoraram mais na segunda seleção, os valores em cada seleção são bem próximos (a diferença máxima foi de 3 minutos), e essas pessoas já teriam obtido um tempo muito baixo na primeira seleção (em média 38,5 minutos). Essas 4 pessoas estão divididas em todos os 4 grupos definidos no estudo, o que indica não existir um comportamento padrão para se obter um melhor tempo na primeira seleção.

Na instância GRAD, dentre os 29 participantes que finalizaram o estudo, 20 realizaram a segunda seleção com tempo inferior à primeira. A distância entre o tempo gasto na primeira seleção e na segunda chega a ser de 100 minutos. Já em relação aos 9 participantes que demoraram mais na segunda seleção, os valores em cada seleção são bem próximos (a média das diferenças ficou em 10 minutos). Essas 4 pessoas estão divididas em todos os 4 grupos definidos no estudo, o que indica não existir um comportamento padrão para se obter um melhor tempo na primeira seleção.

5.6.4 Análise Qualitativa do Aspecto Usabilidade

Ao total, 25 problemas diferentes foram relatados pelos participantes da instância POS e 15 problemas, na instância GRAD. Eles totalizam 30 problemas, pois alguns são repetidos entre as instâncias, e estão associados, em sua maioria, à falta de explicação sobre o que representa cada atributo de caracterização, ausência da padronização dos possíveis valores nos atributos e sugestão de um apoio semi-automatizado que possibilite filtrar TTBMs que não se encaixam ao projeto e que possibilite cruzar suas características (Tabela 5.34).

Esses aspectos podem ser contornados através da construção de um apoio ferramental que padronize os possíveis valores dos atributos de caracterização e filtre as técnicas de acordo com a caracterização do projeto de software.

Na instância POS, *Porantim* obteve menos relato de problemas que o Esquema de Caracterização, mas na instância GRAD o comportamento foi inverso. Uma justificativa para uma quantidade menor de problemas no total na instância GRAD pode se dar pelo fato do treinamento feito para estes participantes ter sido realizado após a instância POS, o que de certa forma o tornou uma evolução natural do primeiro treinamento.

Tabela 5.34. Problemas relatados durante o estudo

ID	Problemas relatados	POS	GRAD
1	Falta de explicação sobre o atributo Critério de Geração dos Testes	5	-
2	Falta de explicação sobre o atributo Resultado dos Testes	4	6
3	Formalizar as opções no campo Paradigma de Desenvolvimento	3	-
4	Indicação do custo da ferramenta (alto, médio ou baixo)	3	-
5	Falta de informação do conhecimento da equipe sobre a Linguagem e sobre o Modelo.	3	1
6	Ausência de uma priorização dos atributos (pesos)	3	-
7	Dificuldade no entendimento do gráfico de adequação	2	2
8	Formalizar as opções no campo Linguagem de Programação	2	1
9	Falta de explicação sobre o atributo Tempo de Desenvolvimento	2	-
10	Ausência do campo: cobertura dos testes	2	-
11	Uma escala de complexidade e qualidade relacionada ao critério de geração dos testes poderia ser informada	2	-
12	Informação de dependência não foi útil	2	-
13	Falta de explicação sobre o atributo Plataforma de Execução	2	8
14	Falta de explicação sobre o atributo Ferramenta de Apoio	2	-
15	Formalizar quais atributos são impeditivos	1	3
16	Indicar quais as ferramentas da empresa	1	-
17	Dificuldade em relacionar atributos de caracterização com as características do projeto	1	3
18	Necessidade de modelagem intermediária não pose ser estimada	1	-
19	Falta de explicação sobre o atributo Paradigma de Desenvolvimento	1	-
20	Falta de explicação sobre o atributo Mecanismo de Rastreabilidade	1	1
21	Falta de explicação sobre o atributo Verificador de Modelo	1	1
22	Falta de explicação sobre os atributos Complexidade dos passos não-automatizados	1	1
23	Falta de explicação sobre o atributo Custo da Ferramenta	1	-
24	Compreensibilidade não foi útil	1	-
25	Alguns campos precisam ser pré-definidos	1	-
26	Dúvida a respeito da entrada requerida e sua relação com o modelo adotado	-	4
27	Falta de explicação sobre o atributo Entrada Requerida	-	1
28	Falta de explicação sobre o atributo Critério de Geração dos Testes	-	1
29	Falta de explicação sobre o atributo Habilidades Requeridas	-	1
30	A forma de visualização dos tipos de falha dificulta a leitura da caracterização	-	1

5.6.5 Análise Qualitativa do Aspecto Satisfação do Usuário

- **Quais são as vantagens e desvantagens em usar *Porantim*?**

Vantagens: Os participantes que usaram ambas as abordagens de seleção preferiram *Porantim* por 2 (duas) razões:

- *Porantim* oferece mais informações para caracterizar TTBMs.
- *Porantim* apresenta um valor numérico sugerindo a adequação de cada TTBM para um projeto de software. Ele possibilita a ordenação de TTBMs antes de iniciar a análise e seleção de TTBMs para um projeto de software.

Desvantagens: elas estão relacionadas ao grande número de atributos de caracterização provido para cada TTBM, o que pode tornar a atividade de seleção uma tarefa exaustiva se existirem muitas TTBMs disponíveis a serem analisadas para um projeto de software.

- **Você usaria *Porantim* em um Ambiente Industrial?**

Os participantes que usaram a versão completa de *Porantim* avaliaram a tecnologia como simples de ser usada. No entanto, dois pontos principais foram reportados:

  - Foi relatado que manter as informações sobre as TTBMs seria uma tarefa complexa, pois elas evoluem rapidamente ao longo do tempo. E isso poderia resultar em muito custo em uma organização.
  - É importante existir um treinamento mais detalhado sobre as funcionalidades providas por *Porantim*. O primeiro uso da abordagem não foi tão simples e foi necessário um certo tempo para entender o significado das informações que existem no corpo de conhecimento de TTBMs.

- **Quais melhorias poderiam ser feitas em *Porantim*?**

Um pequeno número de melhorias foi sugerido, possivelmente justificado pela baixa experiência dos participantes dos estudos na área de teste de software e principalmente em TBM. Essas melhorias foram classificadas em três categorias:

  - Novos atributos de caracterização de TTBMs (ex: esforço requerido para usar uma TTBM);
  - Novos atributos de caracterização para projeto de software (ex: o grau de experiência da equipe de teste ou seu conhecimento em uma tecnologia);
  - A padronização dos possíveis valores para alguns atributos de caracterização (ex: plataforma de execução de software, paradigma de desenvolvimento e linguagem de programação) para evitar que dados livres sejam inseridos e dificultem o processo de seleção. Esta foi a sugestão mais frequente.

Todas as sugestões foram processadas na versão final da abordagem *Porantim*, descrita no Capítulo 4.

- **Você mudou a forma como vê o problema de seleção de tecnologias após usar *Porantim*? O que você aprendeu usando *Porantim*?**

Todos os participantes que usaram *Porantim* relataram que mudaram a forma de ver tal problema da seleção de tecnologias. No entanto, é importante lembrar que nenhum participante possuía experiência anterior na seleção de TTBMs para projetos de software.

Além disso, todos eles reportaram que aprenderam a observar quais características ou informações seriam realmente importantes ou não quando estão

selecionando tecnologias para um projeto de software e como combinar a análise dessas características para a tomada de decisão.

5.7 Considerações Finais do Capítulo

Este Capítulo apresentou um estudo experimental conduzido para avaliar a abordagem de apoio à seleção de TTBMs proposta neste trabalho (*Porantim*) em relação a outra abordagem de apoio à seleção de técnicas de teste.

O estudo foi realizado com duas amostras com perfis diferentes, alunos de pós-graduação e alunos de graduação, e seus resultados indicam que, dada a população do estudo e os tipos de projeto de software utilizados, *Porantim* poderia contribuir para melhoria na efetividade e eficiência do processo de seleção de TTBMs em projetos de software com diferentes características, quando comparada à abordagem Esquema de Caracterização proposta por VEGAS e BASILI (2005). No entanto, os resultados não são conclusivos em virtude do pouco volume de dados obtidos.

A abordagem *Porantim* foi ainda avaliada subjetivamente pelos participantes do estudo a respeito da sua satisfação durante o uso da abordagem, e seus resultados também foram bastante positivos dentro da população do estudo. Ao longo do estudo, os participantes sugeriram por diferentes razões a construção de um apoio computacional para auxiliar no uso da abordagem *Porantim*.

Uma possibilidade para este estudo seria agrupar os resultados, juntando os dados obtidos nas duas amostras a fim de analisar o comportamento da abordagem *Porantim* com um grupo mais heterogêneo. No entanto, isso não foi realizado neste trabalho, pois houve mudanças no treinamento das abordagens em cada instância e isso poderia influenciar nos resultados a serem obtidos.

Após a avaliação da abordagem *Porantim*, percebeu-se que poderia ser difícil a aplicação manual das fórmulas metamáticas que compõem *Porantim* para todos os projetos onde se desejam aplicar TTBMs. Sendo assim, sugere-se que essas fórmulas sejam implementadas em planilhas eletrônicas ou através de um apoio ferramental para que sejam calculadas automaticamente. Um apoio computacional foi desenvolvido para apoiar os cálculos a serem efetuados durante o uso de *Porantim* e na representação das informações geradas através de interfaces visuais e gráficos que auxiliam na análise e tomada de decisão a respeito da seleção de TTBMs. Este apoio será descrito no Capítulo seguinte.

CAPÍTULO 6 - INFRA-ESTRUTURA COMPUTACIONAL DE APOIO À SELEÇÃO DE TTBMS

Neste capítulo é apresentada a infra-estrutura computacional desenvolvida nesta pesquisa, seus requisitos, arquitetura e funcionalidades, para apoiar a seleção de TTBMs seguindo a abordagem Porantim.

6.1 Introdução

Durante a definição e avaliação da abordagem de seleção *Porantim*, foi observada a possibilidade de automatizar alguns passos que compõem o processo de seleção de TTBMs, pois eles envolvem cálculos aplicando fórmulas matemáticas que são passíveis de erro quando realizados manualmente. Com o propósito de reduzir a possibilidade de erro durante os cálculos associados aos passos do processo de seleção de TTBMs e estruturar o corpo de conhecimento sobre TTBMs, foi desenvolvido um apoio computacional para a abordagem *Porantim*.

Este apoio computacional provido para a abordagem *Porantim* foi desenvolvido como uma extensão da infra-estrutura computacional Maraká, descrita em (DIAS-NETO e TRAVASSOS, 2006; DIAS-NETO, 2006), através da inclusão de novos componentes e funcionalidades que provêem apoio à seleção de TTBMs seguindo o processo que compõe a abordagem *Porantim*.

6.2 A Infra-estrutura Maraká

Maraká é uma infra-estrutura computacional que apóia o planejamento e controle do processo de teste, e foi desenvolvida por DIAS-NETO e TRAVASSOS (2006). Maraká é desenvolvida sobre o framework de uso livre Joomla! (www.joomla.org) e utiliza as tecnologias PHP+MySQL, caracterizando a plataforma de construção da infra-estrutura.

Dentre as principais funcionalidades providas por Maraká, podem ser citadas:

- Execução e acompanhamento do processo de testes e suas atividades para um projeto de software (Figura 6.1);

Figura 6.1. Acompanhamento do Processo de Teste em Maraká

- Geração semi-automática de artefatos do processo de testes de acordo com o Padrão IEEE-829 (Figura 6.2);

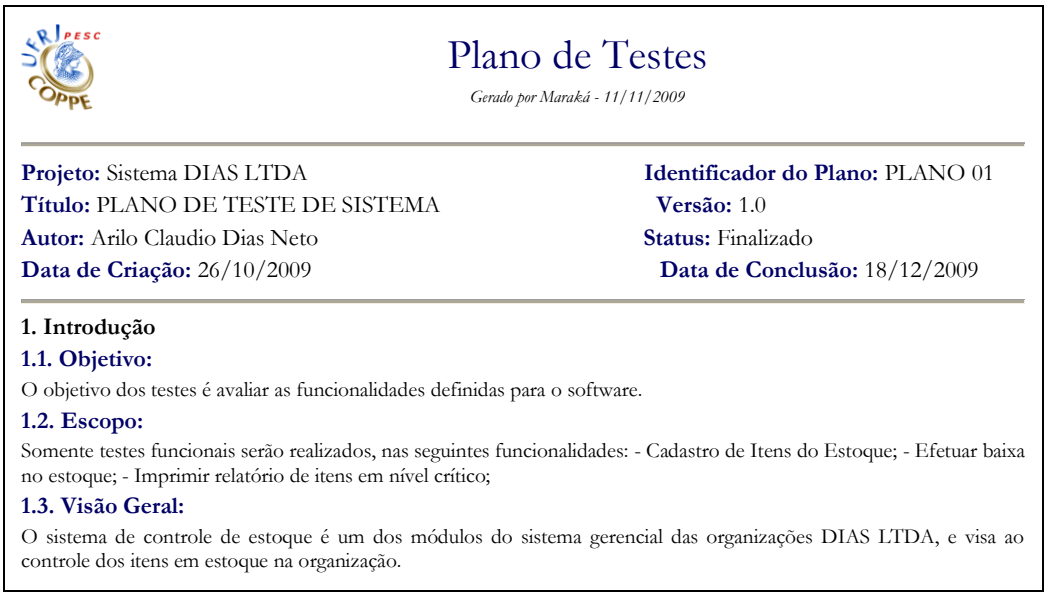


Figura 6.2. Extrato do Plano de Teste gerado por Maraká

- Controle do cronograma e resultados dos testes por meio de gráfico e tabelas (Figura 6.3);

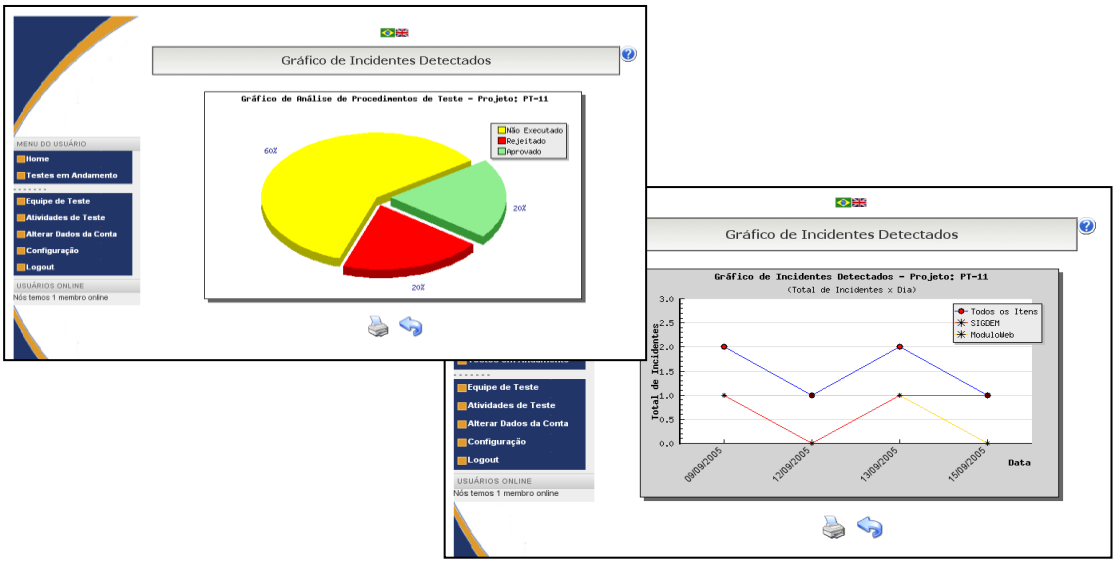


Figura 6.3. Gráficos de Acompanhamento dos Resultados dos Testes

- Gerenciamento dos diferentes papéis envolvidos no processo de teste (Gerente de Teste, Projetista de Teste e Testador), assim como o controle de acesso de cada membro da equipe de teste a um projeto de acordo com seu papel (Figura 6.4).

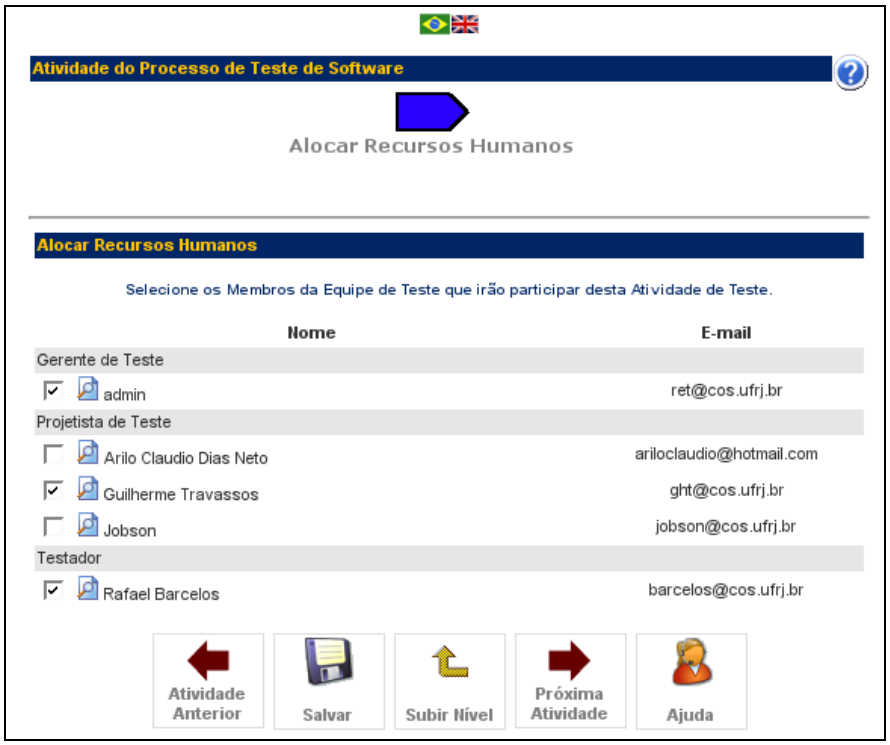


Figura 6.4. Gerenciamento da Equipe de Teste em Maraká

6.3 Evolução da Arquitetura da Infra-estrutura Maraká

A extensão de Maraká para prover apoio ao processo de seleção de TTBMs que compõe a abordagem *Porantim* foi feita com a inclusão do repositório *Técnicas de TBM* e do componente *Gerenciador de Porantim* (destacados na Figura 6.5). Tal componente está diretamente associado ao componente de Maraká responsável pelo gerenciamento do processo de teste para um projeto de software.

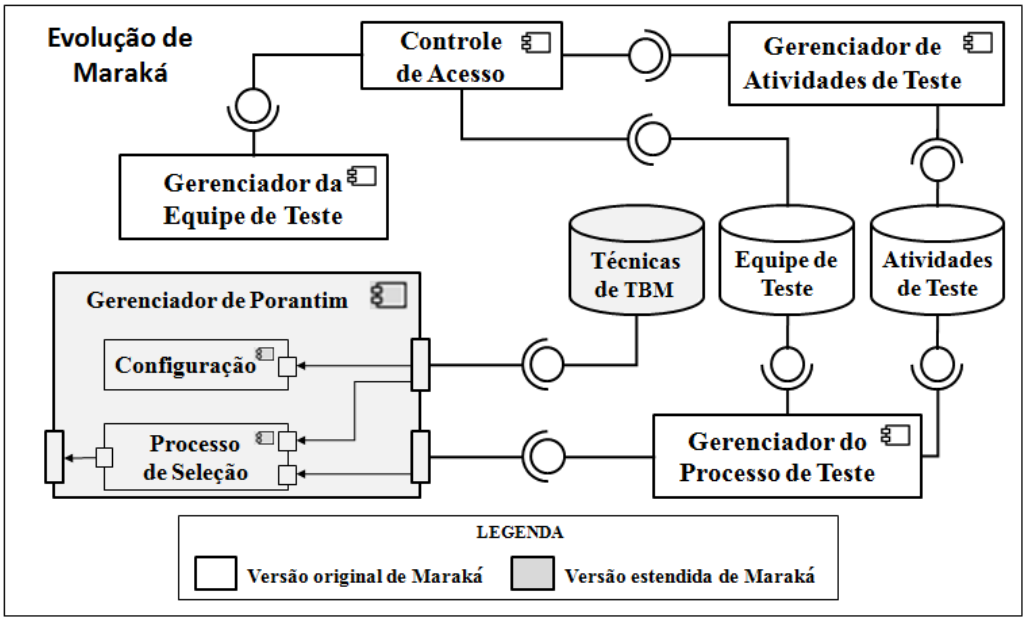


Figura 6.5. Evolução da Arquitetura de Maraká

O detalhamento sobre os demais componentes e repositórios providos por Maraká pode ser obtido em (DIAS-NETO, 2006). Neste trabalho serão detalhados apenas os elementos que estendem a arquitetura de Maraká para prover apoio à abordagem *Porantim*.

O repositório *Técnicas de TBM* implementa o corpo de conhecimento de TTBMs com a estrutura e conteúdo apresentados no Capítulo 3 deste trabalho, enquanto que o componente *Gerenciador de Porantim* implementa o processo de seleção de TTBMs (apresentado no Capítulo 4 - Seção 4.2), que correspondem aos dois elementos que compõem a abordagem *Porantim*. O componente *Gerenciador de Porantim* é responsável pelas seguintes funcionalidades:

1. Configuração do corpo de conhecimento de *Porantim*, que inclui o gerenciamento (cadastro, edição, exclusão e importação) das TTBMs, acessíveis apenas pelos usuários com perfil de Administrador de Maraká.
2. Configuração dos parâmetros adotados por *Porantim* no processo de seleção de TTBMs, que inclui a definição dos pesos de cada atributo de caracterização

de TTBMs e definição da quantidade máxima de TTBMs a serem exibidas no processo de seleção. Tais funcionalidades são acessíveis apenas pelos usuários com perfil de Administrador de Maraká.

3. Execução do processo de seleção de TTBMs provido por *Porantim*, realizado para cada projeto de teste pelo usuário com o papel Gerente de Teste (responsável pelo planejamento dos testes utilizando a infra-estrutura Maraká).

A seguir serão descritas as funcionalidades implementadas no componente *Gerenciador de Porantim* para apoiar o uso da abordagem *Porantim* através da infra-estrutura Maraká. Esta versão da infra-estrutura Maraká pode ser acessada através da URL: http://lens-ese.cos.ufrj.br/*Porantim*.

6.4 Funcionalidades

Nesta Seção, as funcionalidades que compõem o componente *Gerenciador de Porantim* serão apresentadas de acordo com as três responsabilidades deste componente, conforme listado na Seção anterior.

6.4.1 Apoio à Configuração do Corpo de Conhecimento de *Porantim*

A configuração do corpo de conhecimento de *Porantim* envolve a disponibilização do repositório de TTBMs com seus atributos de caracterização. Dessa forma, é possível cadastrar, consultar, editar, excluir TTBMs, além de configurar os pesos de cada atributo de caracterização de TTBMs usado por *Porantim* durante o processo de seleção. Para acessar esta funcionalidade, a partir da tela inicial de Maraká escolha a opção Configuração → Técnicas de TBM.

Ao escolher esta opção, será exibida a tela de consultas das TTBMs incluídas no Repositório de TTBMs, exibindo suas principais características (Figura 6.6).

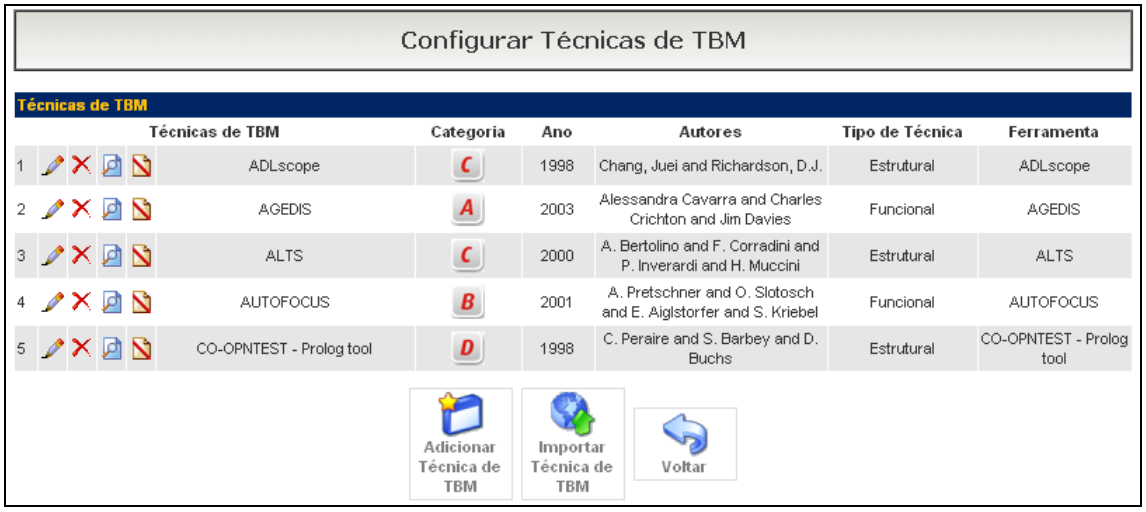


Figura 6.6. Tela de Consulta de TTBMs incluídas no repositório

A partir desta tela, o usuário pode acessar as funcionalidades para adicionar uma nova TTBM no repositório (Figura 6.7), editar os dados de uma TTBM já cadastrada, visualizar os dados de uma TTBM inserida no repositório, excluir uma TTBM do repositório ou importar dados de novas TTBMs a partir da ferramenta JabRef (esta funcionalidade será descrita com maiores detalhes na Seção 6.5.

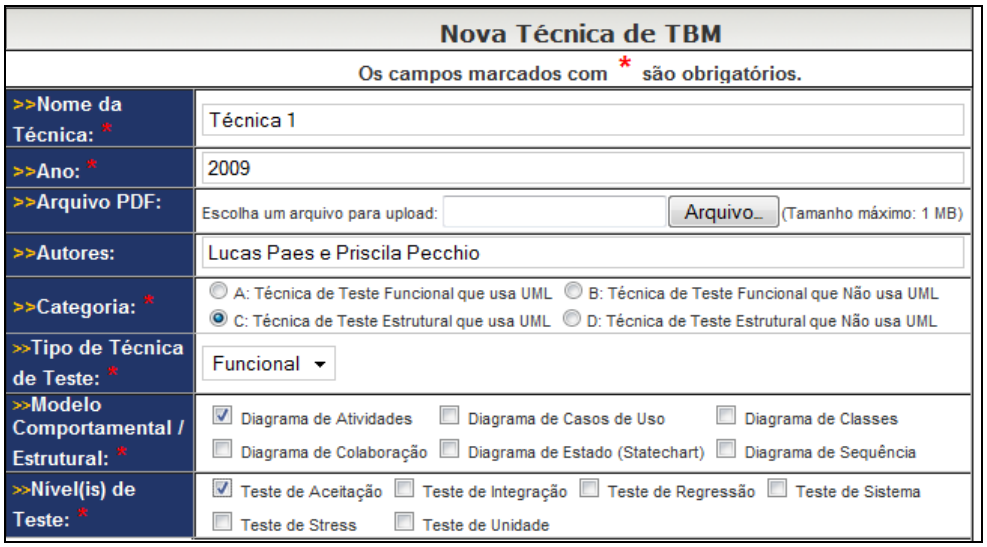


Figura 6.7. Tela Parcial de Cadastro de uma nova TTBM

6.4.2 Apoio à Configuração dos Parâmetros de *Porantim*

Outra funcionalidade provida por Maraká permite ajustar o peso dos atributos de caracterização de TTBMs previamente definidos para o processo de seleção. Tais pesos foram definidos inicialmente através de um *survey* publicado em (DIAS-NETO e TRAVASSOS, 2008) e descrito no Capítulo 3 - Seção 3.3, porém podem ser alterados ao longo do tempo de acordo com as características e necessidades da organização de software que esteja utilizando a abordagem *Porantim*. A soma dos pesos de todos os atributos de caracterização deve ser igual a 1, representando 100%. Também pode ser configurado se o atributo em questão poderá ser um atributo restritivo do projeto de software, ou seja, poderá filtrar TTBMs apenas que atendam obrigatoriamente à característica do projeto associada a este atributo. Isso é definido através as seleção do campo OPCIONALIDADE localizado ao lado do campo de peso de cada atributo de caracterização (ver Figura 6.8).

Ainda é possível fixar a quantidade máxima de TTBMs a serem sugeridas pela infra-estrutura após o cálculo do grau de adequação entre as técnicas e o projeto, e determinar se um atributo deve ser obrigatório no processo de seleção (Figura 6.8).

Para acessar esta funcionalidade, a partir da tela inicial de Maraká escolha a opção Configuração → Configurar *Porantim*.

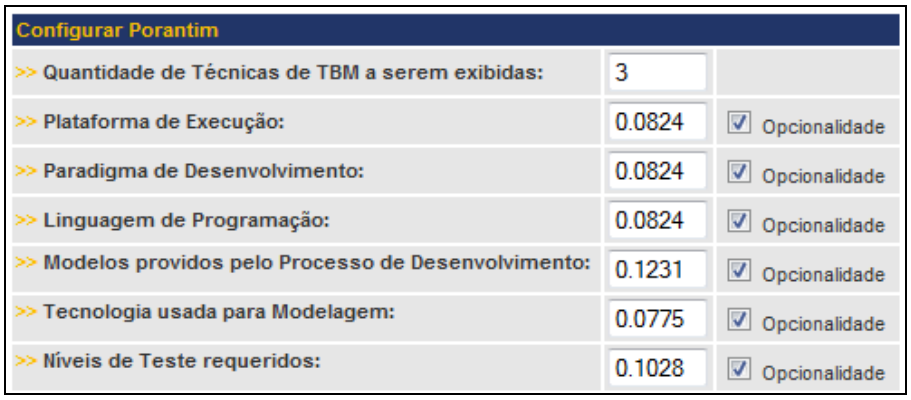


Figura 6.8. Tela de Configuração dos Atributos de Caracterização de TTBM e seus pesos

6.4.3 Apoio ao Processo de Seleção de Técnicas de TBM

A seleção de técnicas de teste para um projeto de software normalmente ocorre durante a atividade de Planejamento dos Testes. Nesse contexto, a infra-estrutura Maraká possui um processo de teste contendo a atividade *Planejar Testes* e uma sub-atividade chamada *Definir Técnicas de Teste*. Na versão original de Maraká, esta atividade é feita através da descrição manual das técnicas de teste que serão adotadas. Já em sua extensão para apoiar a abordagem *Porantim*, é permitida a opção entre definir técnicas de teste de forma manual ou com o auxílio de *Porantim*, lembrando que *Porantim* só se aplica ao contexto de TTBMs. Esta funcionalidade implementa o elemento "Processo de Seleção", apresentado na Seção 4.2.

Para acessar a atividade de seleção de técnicas de teste foi criado um atalho que pode ser acessado a partir da tela inicial de Maraká através do caminho: Testes em Andamento → Selecionar um Projeto (clicar no ícone Abrir para um projeto disponível) → Atalho para Seleção de Técnicas de Teste (Figura 6.10).

Figura 6.9. Atalho para tela de Seleção de Técnicas de Teste

Para realizar a seleção de técnicas de teste seguindo o processo definido por *Porantim* é preciso escolher a abordagem de seleção *Porantim* e então clicar no botão "Iniciar o uso de *Porantim*" (Figura 6.10).

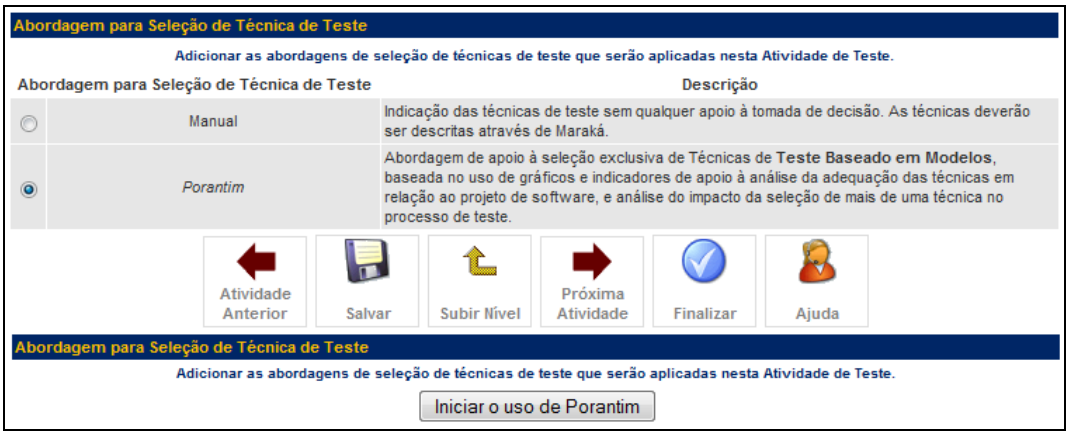


Figura 6.10. Tela de Escolha da Abordagem para Seleção de Técnicas de Teste em Maraká

Feito isso, a execução será direcionada à funcionalidade responsável pelo apoio à seleção de TTBMs. Tal funcionalidade foi dividida em três passos:

6.4.3.1 PASSO 1) Caracterização do Projeto de Software

Atividade na qual a equipe de teste define as características e requisitos de teste para o projeto de software em que se deseja aplicar TTBMs através de um formulário (Figura 6.11) contendo os atributos de caracterização de projeto de software apresentados na Tabela 4.1.

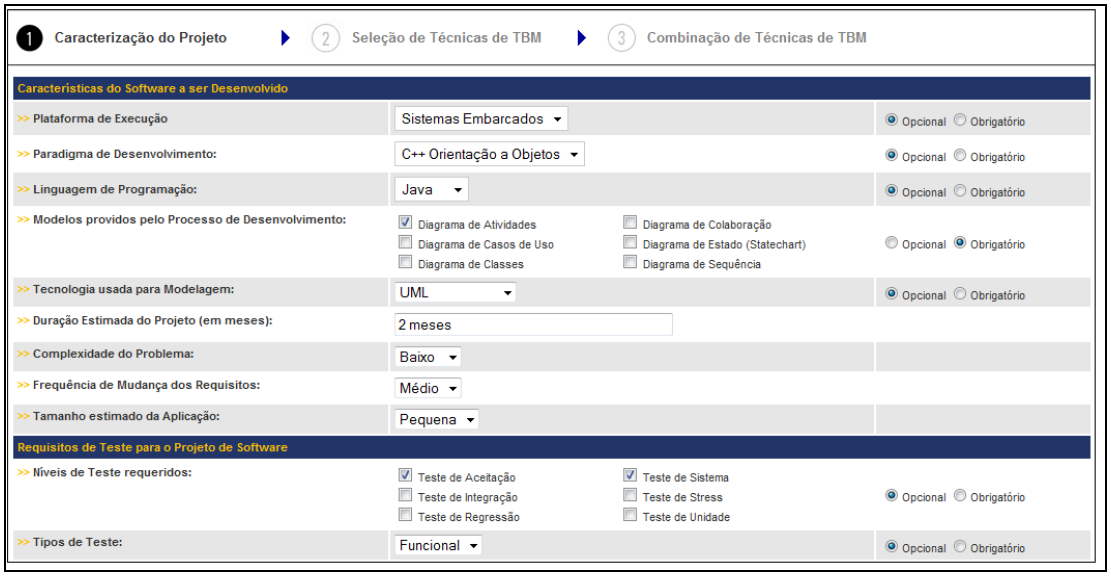


Figura 6.11. Tela de Caracterização do Projeto de Software

6.4.3.2 PASSO 2) Seleção de Técnicas de TBM

Atividade onde são calculados automaticamente os graus de adequação de todas as TTBMs existentes no repositório em relação ao projeto de software

caracterizado no Passo 1, seguindo as fórmulas providas por *Porantim* (apresentadas no Capítulo 4 - Seção 4.2.2). Em seguida, as técnicas são exibidas ordenadas de forma decrescente por grau de adequação. A quantidade máxima de técnicas disponibilizadas para seleção corresponde ao valor definido na tela de configuração (Figura 6.8). Para cada técnica exibida, a infra-estrutura provê uma funcionalidade (um link) que fornece uma análise detalhada da adequabilidade da TTBM selecionada em relação ao projeto caracterizado.

Como sugestão de representação gráfica da adequação de uma TTBM a um projeto de software, adotou-se o uso de um *Gráfico de Radar* (Figura 6.12). Um gráfico de radar corresponde a um método gráfico para exibir dados com múltiplos valores na forma de um gráfico bi-dimensional de três ou mais variáveis quantitativas representadas nos vértices, iniciando a partir de um mesmo ponto. Em *Porantim*, as variáveis correspondem aos atributos de caracterização de projetos de software/TTBMs. Com isso, ele possibilita a visualização gráfica da adequação de uma TTBM a um projeto de software.

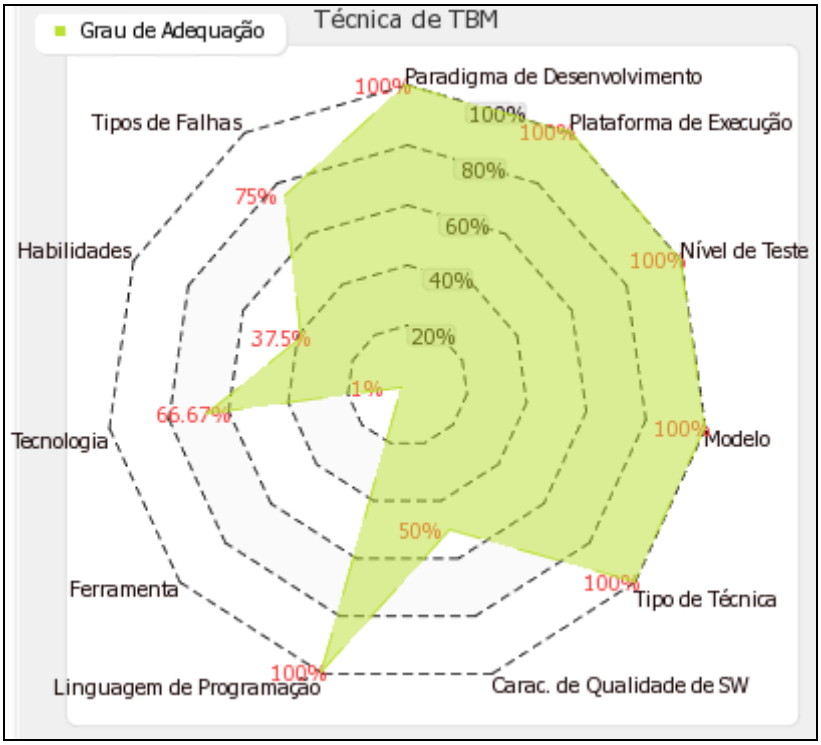


Figura 6.12. Tela de Análise da Adequação de TTBM via Gráfico de Radar

Neste passo, a equipe de teste deve selecionar as TTBMs a serem adotadas no projeto. Este passo implementa as atividades 2, 3 e 4 do elemento “Processo de Seleção” (Figura 4.1).

6.4.3.3 PASSO 3) Combinação de Técnicas de TBM

Atividade onde é realizada a análise do impacto das TTBMs selecionadas no Passo 2 a partir do cálculo de três indicadores, citados na Seção 4.2.4 deste trabalho: *Cobertura do Projeto de Software*, *Esforço de Modelagem Salvo* por utilizar modelos já providos pelo Processo de Desenvolvimento e *Grau de Preparação da Equipe de Teste* selecionada para o Projeto.

As informações podem ser visualizadas a partir de uma tabela que lista os dados do projeto e das técnicas selecionadas, e através de uma representação gráfica.

Conforme descrito na Seção 4.2.4, tais indicadores representam um número entre 0 e 100%. Para simplificar a representação de cada indicador graficamente, seus valores foram divididos em quatro diferentes intervalos: [100-75%]: Alto; [75-50%]: Médio; [50-25%]: Baixo; [25-0%]: Muito Baixo. Em seguida, adotou-se o uso de um *Gráfico de Gauge* para representação gráfica dos indicadores de análise de combinação de TTBMs (Figura 6.13[17]). A escolha de Gráfico de Gauge ocorreu por este tipo de gráfico visualizar facilmente diferentes faixas de valores representando categorias que são separadas por limites pré-estabelecidos. O marcador do "velocímetro" indica quanto o valor obtido atualmente está posicionado em relação a uma meta definida.

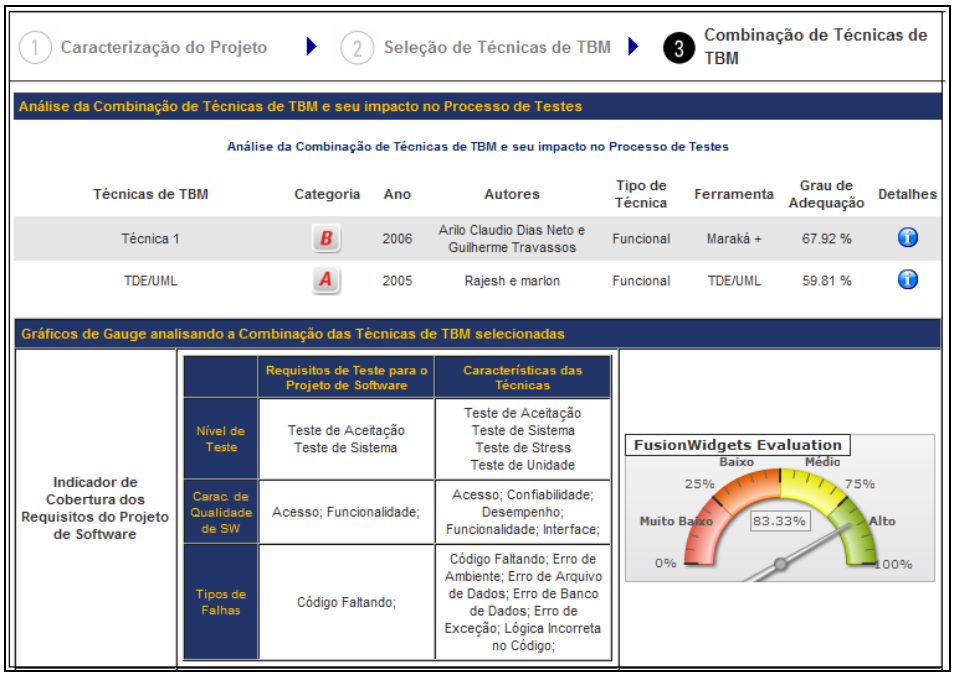


Figura 6.13. Tela de Análise da Combinação de Técnicas de TBM

[17] Para exibição deste gráfico, está sendo utilizada uma versão de avaliação do componente FusionWidgets v3, disponível em http://www.fusioncharts.com/widgets/.

Caso a equipe de teste confirme a escolha, tais técnicas selecionadas serão registradas no plano de teste construído para o projeto em questão e o fluxo de execução retorna às demais atividades do processo de teste monitoradas por Maraká.

6.5 Interface para Manutenção do Corpo de Conhecimento

Como um mecanismo para facilitar a manutenção do corpo de conhecimento de TTBMs provido por Maraká, foi desenvolvida uma facilidade adicional que possibilita a importação de dados de TTBMs a partir de uma das principais fontes de TTBMs: a literatura técnica.

Esta facilidade possibilita que TTBMs caracterizadas a partir de uma ferramenta de gerenciamento de referências bibliográficas (JabRef – http://jabref.sourceforge.net) tenham seus dados importados para o repositório de TTBMs mantido por Maraká. Para isso, foi criada uma interface para carregamento (*upload*) do arquivo no formato de JabRef (.bib) contendo os dados das TTBMs e processamento do arquivo carregado para inclusão de novas TTBMs. No entanto, esta funcionalidade requer que o arquivo de JabRef carregado em Maraká siga um formato pré-definido, contendo um conjunto definido de *tags* (rótulos) descrevendo os atributos de caracterização de uma TTBM.

As etapas para configuração de JabRef para possibilitar a importação de TTBMs e a funcionalidade de apoio à importação de TTBMs provida por Maraká serão descritas a seguir:

- **Configuração de JabRef**

1. O primeiro passo para configuração de JabRef é a definição dos campos que irão compor o formulário de caracterização de uma referência bibliográfica, que irá representar uma TTBM. Para isso, utilizando JabRef, acesse o menu "*Options → Set up General Fields*", que resultará na tela apresentada na Figura 6.14. Nesta tela, as 5 últimas linhas apresentadas não fazem parte da configuração padrão de JabRef e devem ser incluídas para possibilitar a importação de TTBMs em Maraká.

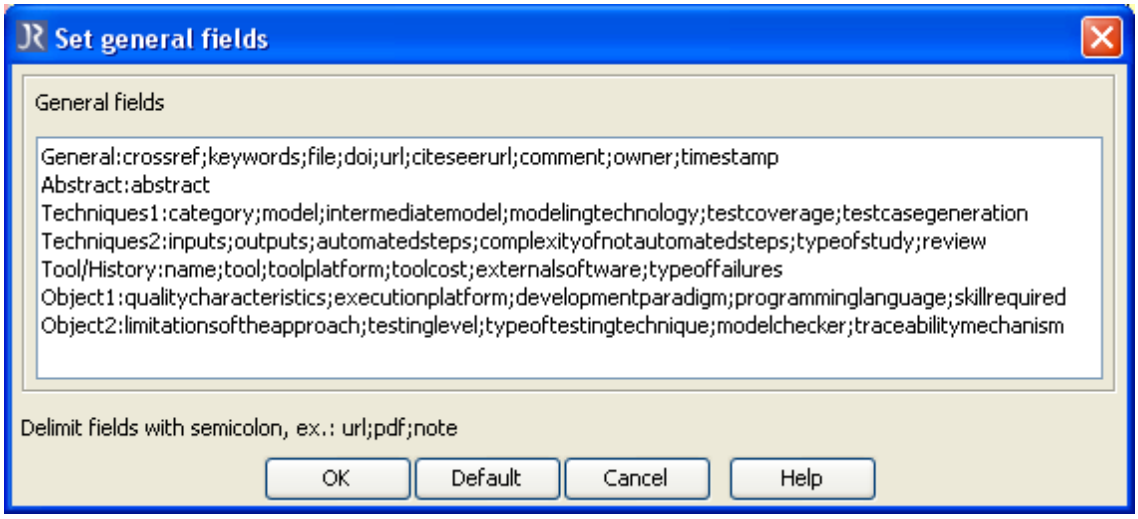


Figura 6.14. Configuração dos campos de caracterização e TTBMs em JabRef

2. Em seguida, devem ser respeitadas algumas regras para preenchimento dos campos de caracterização de TTBMs em JabRef, seguindo o mapeamento e regras definidos na Tabela 6.1.

Tabela 6.1. Regras de Mapeamento e Preenchimento dos dados de TTBMs

Atributo de Caracterização em Maraká	Campo [tag] equivalente em JabRef	Regra de Preenchimento
Nível de automação	Automatedsteps	Escolher entre as opções: *Automated*, *Not Automated* ou *Needs of Intermediate Modeling*
Categoria	Category	Escolher entre as opções A, B, C, D ou E
Complexidade dos passos não automatizados	Complexityofnotautomatedsteps	Formato → "[complexidade]:[descrição]". [complexidade] deve ser escolhida entre as opções *Low*, *Medium*, *High* e *Very High*; e [descrição] pode ser preenchido com qualquer texto.
Paradigma de Desenvolvimento	Developmentparadigm	Cada paradigma deve ser separado por vírgula ","
Plataforma de Execução de Software	Executionplatform	Cada plataforma deve ser separada por vírgula ","
Entradas Requeridas	Inputs	Pode ser preenchido com qualquer texto
Modelo Intermediário	intermediatemodel	Pode ser preenchido com qualquer texto
Limitações	Limitationsoftheapproach	Pode ser preenchido com qualquer texto
Modelo Comportamental/Estrutural	Model	Cada modelo deve ser separado por vírgula ","
Existência de um Verificador de Modelo	Modelchecker	Escolher entre as opções YES ou NO
Tecnologia usada para Modelagem	Modelingtechnology	Pode ser preenchido com qualquer texto
Nome da Técnica	Name	Pode ser preenchido com qualquer texto
Resultados Gerados	Outputs	Pode ser preenchido com qualquer texto
Linguagem de Programação	Programminglanguage	Cada linguagem deve ser separada por vírgula ","
Características de Qualidade de Software	Qualitycharacteristics	Cada característica deve ser separada por vírgula "," (ex: functionality, security)
Habilidades requeridas para ser operada	Skillrequired	Cada habilidade deve ser separada por vírgula ","
Critério de Geração dos Testes	Testcasegeneration	Pode ser preenchido com qualquer texto
Critério de Cobertura	Testcoverage	Pode ser preenchido com qualquer texto
Nível de Teste	Testinglevel	Escolher entre as opções *Acceptance Testing, Integration Testing, Regression Testing, System Testing, Unit Testing*
Ferramenta de apoio	Tool	Pode ser preenchido com qualquer texto

Custo da Ferramenta	Toolcost	Formato → "[custo]:[descrição]". [custo] deve ser escolhido entre as opções *Freeware*, *Shareware*; e [descrição] pode ser preenchido com qualquer texto.
Plataforma em que a ferramenta opera	Toolplatform	Pode ser preenchido com qualquer texto
Existência de um Mecanismo de Rastreabilidade	Traceabilitymechanism	Escolher entre as opções YES ou NO
Tipos de falhas que permitem revelar	Typeoffailures	Cada falha deve ser separada por vírgula ","
Tipo de Estudo	Typeofstudy	Escolher entre as opções *Case Study, Experimental Study, Industrial Experience Report, Proof of Concept, Theoretical*
Tipo de Técnica de Teste	Typeoftestingtechnique	Escolher entre as opções *Functional* ou *Structural*
Ano	Year	Pode ser preenchido com qualquer ano
Autores	Author	Pode ser preenchido com qualquer texto
Artigo	Title	Pode ser preenchido com qualquer texto
Arquivo PDF	File	Preenchido com o caminho do arquivo PDF na máquina de quem está fazendo a importação
Referência Completa	Combinação dos campos "autor+title+booktitle ou journal+year+pages".	Cada campo pode ser preenchido com qualquer valor

Com isso, teremos um arquivo de JabRef no formato .bib apto a ser importado por Maraká.

- **Importando TTBMs com Maraká**

Para utilizar esta funcionalidade, a partir da tela de consulta de TTBMs (Figura 6.6) devemos clicar na opção "Importar Técnica de TBM", que redirecionará Maraká à tela apresentada na Figura 6.15.

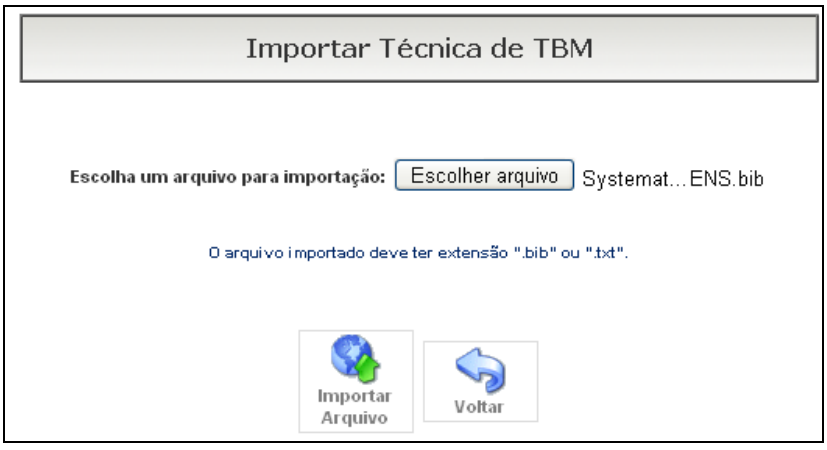


Figura 6.15. Tela para upload do arquivo a ser importado por Maraká

Em seguida, Maraká apresenta uma tela listando todas as TTBMs identificadas a partir do arquivo selecionado. Neste momento, o usuário pode escolher entre quais TTBMs deseja importar para o repositório de TTBMs em Maraká (Figura 6.16).

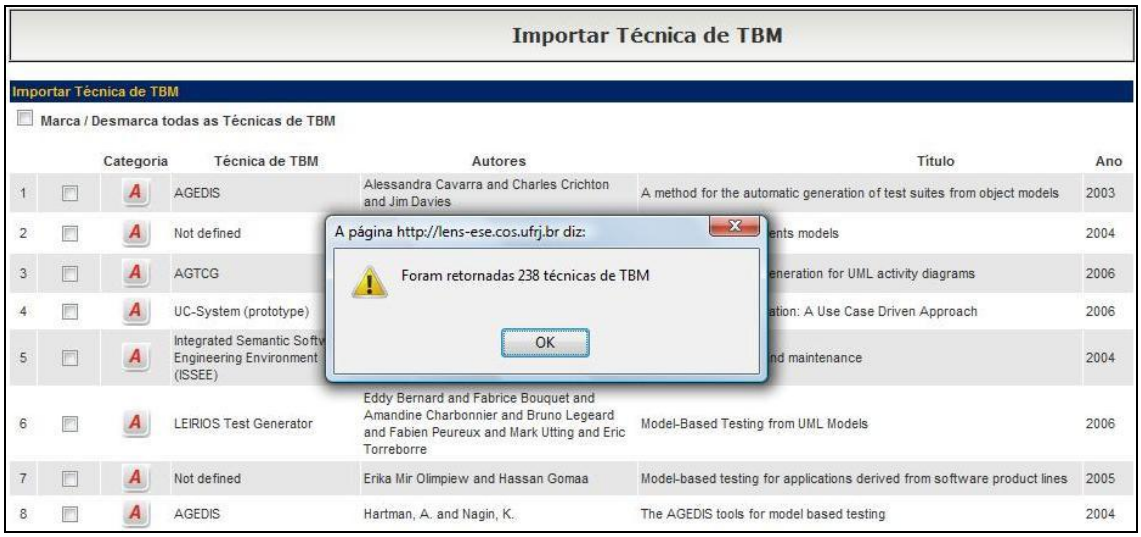


Figura 6.16. Tela para seleção das TTBMs a serem importadas por Maraká

Após a confirmação, as TTBMs selecionadas terão seus dados armazenados no repositório de TTBMs em Maraká e estarão aptas a fazerem parte do processo de seleção de TTBMs para um projeto de software.

6.6 Avaliação da Infra-Estrutura Computacional

Para avaliação da infra-estrutura computacional desenvolvida para apoiar o processo de seleção de TTBMs provido por *Porantim* foi realizado um estudo de avaliação em uma organização de software sediada nos EUA que utiliza TBM em seus projetos.

6.6.1 Planejamento do Estudo

6.6.1.1 Definição do Objetivo Global

O objetivo deste estudo é avaliar as funcionalidades providas pela infra-estrutura computacional de apoio à abordagem *Porantim*, analisando o comportamento e viabilidade da infra-estrutura desenvolvida em relação à seleção provida por especialistas em TBM. Serão avaliadas: (1) a indicação de um subconjunto de TTBMs mais adequadas a um projeto e (2) a geração de indicadores analisando o impacto da combinação de mais de uma TTBM selecionada para um mesmo projeto que apóiam na seleção.

6.6.1.2 Objetivo Detalhado do Estudo

Analisar as funcionalidades providas por *Porantim* e seu apoio ferramental

Com o propósito de caracterizar

Com respeito à efetividade, eficiência e usabilidade na indicação de TTBMs e geração de indicadores de impacto da combinação de TTBMs

No ponto de vista de engenheiros de software que atuam com TTBMs

No contexto de projetos de software pré-caracterizados.

6.6.1.3 Questões de Pesquisa

A Tabela 6.2 descreve os diferentes aspectos que serão avaliados sobre *Porantim* durante o estudo.

Tabela 6.2. Questões a serem respondidas durante o experimento

Aspecto	Questões
Eficiência	• Quanto tempo leva a seleção de TTBMs usando o apoio ferramental provido por *Porantim*? E sem tal apoio? • Quanto tempo leva para decidir se uma TTBM será usada ou não em um projeto usando o apoio ferramental provido por *Porantim*? E sem tal apoio?
Efetividade	• *Porantim* indica as TTBMs mais adequadas a um projeto de software? Ou seja, os participantes concordam com as técnicas descartadas por *Porantim*? • Os indicadores de impacto da combinação de TTBMs providos por *Porantim* são adequados? Ou seja, os indicadores gerados fazem sentido e efetivamente apóiam a seleção de TTBMs? • As funcionalidades providas por *Porantim* direcionam a equipe de teste na escolha combinada de TTBMs que mais se adéquam ao projeto?
Usabilidade	• Quais são as vantagens e desvantagens do uso de *Porantim* e seu apoio ferramental? • Quais melhorias poderiam ser feitas? • O uso do apoio ferramental provido por *Porantim* simplifica o processo de seleção de TTBMs?

6.6.1.4 Medições

Estas medidas devem ser coletadas com e sem a ferramenta construída para apoiar *Porantim*.

Q1: Qual é a eficiência da infra-estrutura computacional?

- **Métrica 1.1:** *tempo* gasto na caracterização do projeto de software.
- **Métrica 1.3:** *tempo* gasto na análise de cada TTBM.
- **Métrica 1.2:** *tempo total* gasto no processo seleção.

Q2: Qual é a efetividade da infra-estrutura computacional?

- **Métrica 2.1:** *TTBM* removida da lista por ser a menos adequada ao projeto.
- **Métrica 2.2:** *TTBMs* selecionadas para o projeto por serem as mais adequadas ao projeto quando combinadas.

6.6.1.5 Fatores e Suas Alternativas

As variáveis cujos valores serão alterados são: o instrumento para seleção e o projeto de software.

- **Abordagem de Seleção**

O fator instrumento para seleção terá duas possíveis alternativas: o apoio ferramental provido por *Porantim* e a execução manual (sem qualquer apoio ferramental).

- **Seleção usando a infra-estrutura computacional:** os participantes receberão o repositório de TTBMs através de um apoio ferramental que será responsável por excluir 1 TTBM que é considerada a menos adequada ao projeto, e em seguida os participantes devem selecionar 2 TTBMs que eles escolheriam de forma combinada para os projetos de software aos quais foram alocados utilizando a abordagem *Porantim*.
- **Seleção manual:** os participantes receberão o mesmo repositório de TTBMs, porém em um documento impresso. A partir disso eles deverão inicialmente remover 1 TTBM que eles julgam menos adequada ao projeto, e em seguida devem selecionar 2 TTBMs que eles escolheriam de forma combinada para os projetos de software aos quais foram alocados.

Para ambos os métodos serão usadas as mesmas técnicas de TBM.

- **Projeto de Software**

O fator projeto de software inclui todas as características do projeto, incluindo não apenas o software a ser desenvolvido, mas requisitos de teste para o projeto a ser desenvolvido. Neste estudo, foram selecionados dois projetos de software já utilizados no estudo experimental descrito no Capítulo 5:

- **Gerenciamento de Vídeo-Locadora (V):** Sistema de Informação baseado em banco de dados – o objetivo é construir um sistema de gerenciamento de uma vídeo-locadora.
- **Estacionamento (E):** sistema síncrono – o objetivo é construir a software para controlar a disponibilidade de vagas de estacionamento de carros.

A caracterização detalhada dos projetos de software está apresentada na Tabela 5.1 e Tabela 5.3.

6.6.1.6 Participantes do Estudo

Participaram deste estudo 2 gerentes de teste especialistas em TBM, caracterizados na Tabela 6.3. Observa-se que o Participante 1 foi caracterizado como o menos experiente, enquanto que o Participante 2 seria o mais experiente.

Tabela 6.3. Caracterização dos Participantes do Estudo de Avaliação

Item avaliado	Participante 1 (- experiente)	Participante 2 (+ experiente)
Formação Acadêmica	Doutorado	Doutorado
Anos trabalhando com TBM	4	10
Nº de Projetos usando TBM que participou	2	4
Experiência em TBM	Média	Alta
Já realizou a seleção de TTBMs antes?	Não	Sim

6.6.2 Projeto Experimental

O objetivo primário deste estudo é encontrar qual a influência que o apoio ferramental provido por *Porantim* possui no processo de seleção de TTBMs em um projeto de software, ao comparamos com a execução desta tarefa sem qualquer apoio ferramental. Para isso, a tarefa foi planejada em duas etapas, sendo que a primeira Etapa consiste na seleção de TTBMs sem qualquer apoio ferramental e a segunda Etapa, na seleção de TTBMs usando o apoio ferramental provido por *Porantim*.

Cada participante deverá fazer duas seleções. Como existem dois projetos diferentes, para cada dupla de participantes, o primeiro receberá o Projeto *Vídeo* para a etapa 1 e o Projeto *Estacionamento* para a etapa 2. Já o segundo receberá o Projeto *Estacionamento* para a Etapa 1 e o Projeto *Vídeo* para a Etapa 2, conforme a Tabela 6.4.

Tabela 6.4. Alocação de projetos por dupla de participantes

Participantes	Projetos	
	Vídeo	Estacionamento
Participante 1	Seleção 1	Seleção 2
Participante 2	Seleção 2	Seleção 1

6.6.2.1 Técnicas usadas por Projeto

Para cada projeto, serão disponibilizadas 5 TTBM de diferentes características. A escolha das técnicas foi feita por conveniência pelos pesquisadores, selecionado 5 entre 13 TTBMs adotadas no experimento descrito no Capítulo 5. Foi ainda definido um oráculo por especialistas em TBM indicando quais seriam as 2 TTBMs mais adequadas, quando combinadas, para cada projeto e qual seria a TTBM menos adequada que deveria ser a primeira opção a ser excluída. A lista de TTBMs adotadas neste estudo está descrita na Tabela 6.5. A descrição e caracterização de cada TTBM citada estão disponíveis na Seção AP B.2.

Tabela 6.5. TTBMs utilizadas para cada projeto e Oráculo do Estudo

Projetos	TTBMs disponibilizadas	TTBM a ser excluída	TTBMs mais adequadas
Vídeo	• Técnica 1 – Abdurazik e Offut (2000) • Técnica 2 – Briand e Labiche (2002) • Técnica 4 – Hartmann e Nagin (2000) • Técnica 5 – Kim et al. (1999) • Técnica 11 – Chung et al. (1999)	Técnica 11	Técnicas 1 e 2
Estacionamento	• Técnica 2 – Briand e Labiche (2002) • Técnica 4 – Hartmann e Nagin (2000) • Técnica 10 – Du Bousquet et al. (1999) • Técnica 11 – Chung et al. (1999) • Técnica 12 – Parissis e Vassey (2003)	Técnica 11	Técnicas 10 e 12

Os graus de adequação de cada TTBM para cada projeto utilizando a estratégia *Porantim* estão apresentados na Tabela 6.6. Esses valores apenas serão apresentados para os participantes quando estiverem usando o apoio ferramental provido por *Porantim*. Em negrito na última coluna está destacada a combinação de TTBMs mais adequada para cada projeto segundo dois especialistas em TBM consultados ao longo deste trabalho.

Tabela 6.6. Grau de Adequação das TTBMs usadas no estudo por Projeto

Projetos	Técnicas de TBM disponibilizadas	Técnica descartada	Técnicas a serem selecionadas
Vídeo	• # 2 (GA = 60,01%) • # 4 (GA = 55,83%) • # 1 (GA = 50,68%) • # 5 (GA = 42,74%) • # 11 (GA =13,89%)	Técnica 11	• #2 e #4: 83,33% / 50% / 80% • **#2 e #1: 100% / 54,17% / 75%** • #2 e #5: 66,67% / 50% / 75% • #4 e #1: 94.44% / 45% / 100% • #4 e #5: 66,67% / 50% / 100% • #1 e #5: 77,78% / 41,67% / 100%
Estacionamento	• # 12 (GA =47,98%) • # 10 (GA =47,98%) • # 4 (GA = 25,62%) • # 2 (GA = 23,71%) • # 11 (GA =23,66%)	Técnica 11	• **#12 e #10: 68,89% / 100% / 66,67%** • #12 e #4: 65,56% / 35% / 80% • #12 e #2: 72,22% / 32,14% / 60% • #10 e #4: 82,22% / 35% / 83,33% • #10 e #2: 88,89% / 32,14% / 66,67% • #4 e #2: 72,22% / 27,78% / 80%

6.6.2.2 Passos para seleção de TTBMs

O primeiro passo a ser realizado antes do início do estudo é o preenchimento do Formulário de caracterização do participante (E0), apresentado no Apêndice C – Seção AP C.1.1. Após isso, cada participante do estudo ao iniciá-lo, deverá seguir um conjunto de passos de acordo com a etapa que está realizando (seleção com ou sem apoio ferramental). Esses passos estão descritos a seguir:

- **Passos a serem realizados sem apoio ferramental**

1. O participante receberá um pacote do estudo contendo os seguintes documentos:
    - Formulário de caracterização do projeto já preenchido.
    - Documento descrevendo as 5 TTBMs disponíveis para o projeto.
    - Formulário E1 – Execução do Exercício sem apoio ferramental (disponível no Apêndice C – Seção AP C.1.2).
2. A seguir, o participante deverá analisar os dados do projeto e das 5 TTBMs para indicar 1 das TTBMs a ser descartada logo de início por não se adequar às características do projeto.
    - O tempo gasto para realizar essa tarefa deve ser registrado no Formulário E1.
    - A TTBM descartada deve ser indicada no Formulário E1.
3. Feito isso, o participante deverá escolher, entre as 4 TTBMs restantes, obrigatoriamente 2 para serem selecionadas para o projeto por serem mais adequadas ao projeto quando combinadas, ou seja, elas se complementam.
    - O tempo gasto para realizar essa tarefa deve ser registrado no Formulário E1.
    - As TTBMs selecionadas devem ser indicadas no Formulário E1.
4. Esta fase está conclçuída.

- **Passos a serem realizados com a ferramenta construída para apoiar *Porantim***

1. O participante receberá um pacote do estudo contendo os seguintes documentos:
    - Formulário de caracterização do projeto já preenchido.

- Documento descrevendo as 5 TTBMs disponíveis para o projeto.
- Formulário E2 – Execução do Exercício com o apoio ferramental provido por *Porantim* (disponível no Apêndice C – Seção AP C.1.3).
- Formulário E3 – Formulário de Avaliação do apoio ferramental provido por *Porantim* (disponível no Apêndice C – Seção AP C.1.4).

2. A seguir, o participante deverá preencher a caracterização do projeto na ferramenta e salvar os dados.
   - A ferramenta irá calcular o grau de adequação das técnicas e irá remover automaticamente 1 TTBM.
   - O tempo gasto para realizar essa tarefa deve ser registrado no Formulário E2.

3. Feito isso, o participante através da ferramenta irá para o passo a seguir, onde irá escolher, entre as 4 TTBMs restantes, obrigatoriamente 2 para serem selecionadas para o projeto por serem mais adequadas ao projeto quando combinadas, ou seja, elas se complementam.
   - Para isso, o participante deve consultar as funcionalidades disponibilizadas pelo apoio ferramental: análise individual de adequação de uma TTBM e análise do impacto da combinação de TTBMs para o projeto.
   - O tempo gasto para realizar essa tarefa deve ser registrado no Formulário E2.
   - As TTBMs selecionadas devem ser indicadas no Formulário E2.

4. Por fim, o participante deve analisar as 5 TTBMs disponibilizadas para que ele indique quais delas ele descartaria da seleção no início do processo.
   - A TTBM que seria descartada deve ser indicada no Formulário E2.

5. O participante deve preencher o Formulário E3.

6. O estudo está encerrado.

6.6.3 Análise dos Dados Quantitativos

A Tabela 6.7 apresenta os resultados quantitativos fornecidos por cada participante ao longo do estudo através de um questionário.

Tabela 6.7. Resultados obtidos na avaliação da infra-estrutura computacional

Item avaliado	Participante 1	Participante 2
Fase 1 – Seleção manual de TTBMs		
Projeto usado da seleção manual de TTBMs	Vídeo	Estacionamento
Tempo (min) para caracterização do projeto de software sem apoio computacional	10 minutos	10 minutos
Tempo (min) para seleção manual de TTBMs	15 minutos	5 minutos
Tempo total (min) para execução do processo de seleção	25 minutos	15 minutos
TTBM excluída	Técnica 11	Técnica 11
TTBMs selecionadas	Técnicas 1 e 2	Técnicas 10 e 12
Fase 2 – Seleção de TTBMs com o apoio da infra-estrutura		
Projeto usado da seleção de TTBMs	Estacionamento	Vídeo
Tempo (min) para caracterização do projeto de software sem apoio computacional	6 minutos	5 minutos
Tempo (min) para seleção de TTBMs usando a infra-estrutura	8 minutos	2 minutos
Tempo total (min) para execução do processo de seleção	14 minutos	7 minutos
TTBM excluída	Técnica 11	Técnica 11
TTBMs selecionadas	Técnicas 10 e 12	Técnicas 1 e 2

Apesar de poucos, os dados indicam que o uso da infra-estrutura reduziu à metade o tempo total para caracterização do projeto de software (Figura 6.17), para seleção de TTBMs (Figura 6.18) e, consequentemente, para a execução do processo de seleção de TTBMs (Figura 6.19) independente do projeto usado e do grau de experiência do profissional.

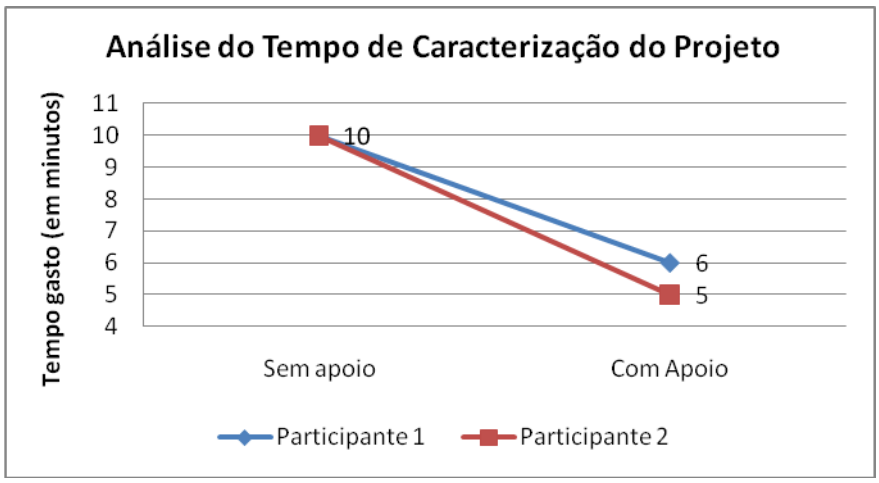


Figura 6.17. Análise do Tempo para Caracterização do Projeto de Software

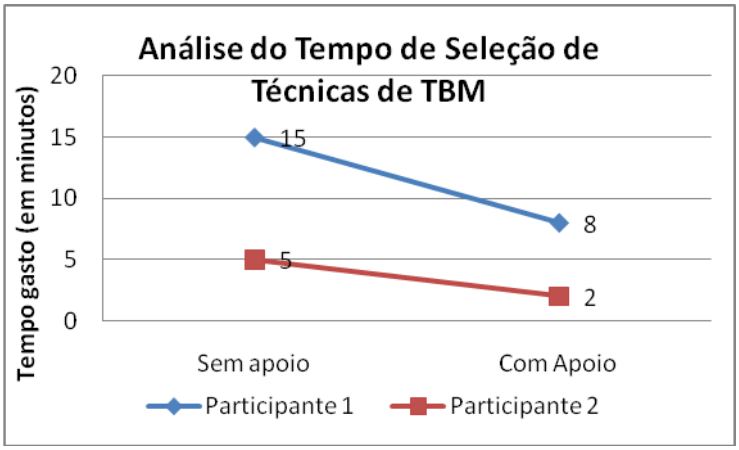


Figura 6.18. Análise do Tempo para Seleção de TTBMs

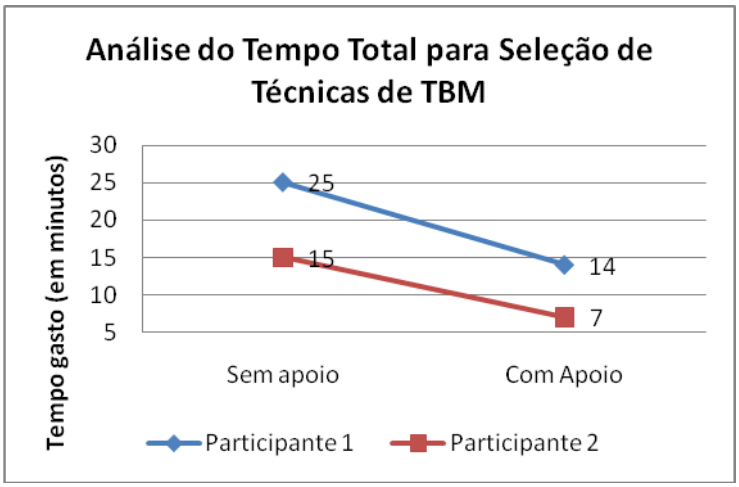


Figura 6.19. Análise do Tempo para Seleção de TTBMs

A análise dos dados por experiência dos participantes permite observar, no gráfico apresentado na Figura 6.20, que o uso do apoio pelo participante menos experiente o aproximou do tempo obtido pelo participante mais experiente para o projeto *Vídeo*. Já quando o mais experiente usou o apoio provido, seu tempo caiu significativamente (projeto *Estacionamento*).

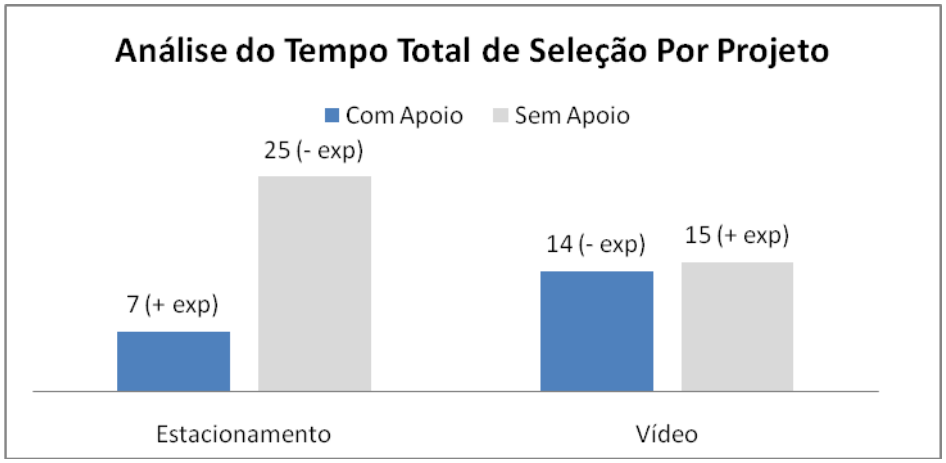


Figura 6.20. Análise por Projeto e Grau de Experiência

Em termos de acertos em relação à exclusão da TTBM que menos se adéqüe ao projeto de software e em relação à seleção das 2 TTBMs que combinadas mais se adéquam ao projeto, os resultados obtidos pelos participantes foram similares ao oráculo pré-definido para o estudo, indicando que o apoio provido por *Porantim* direciona a escolha de TTBMs a um resultado equivalente ao obtido por um especialista em TBM realizando a seleção sem qualquer apoio.

6.6.4 Análise dos Dados Qualitativos

Após a conclusão do exercício, cada participante recebeu um questionário a ser preenchido informando uma análise subjetiva a respeito do uso da infra-estrutura computacional que apóia a abordagem *Porantim*. A análise foi direcionada a partir de algumas perguntas realizadas:

- **Quais os aspectos negativos em usar a infra-estrutura computacional?**

Ambos os participantes indicaram a ausência de um mecanismo que possibilite filtrar TTBMs escolhendo quais atributos de caracterização usar durante o cálculo do grau de adequação de TTBMs. No entanto, esta funcionalidade existe na área de configuração dos parâmetros de *Porantim*, descrita na Seção 6.4.2. Os participantes receberam a infra-estrutura já configurada, por isso não tiveram acesso a esta funcionalidade.

- **Quais os aspectos positivos em usar a infra-estrutura computacional?**

Os participantes relataram que a infra-estrutura é bastante simples de ser usada e provê várias informações que auxiliam na escolha de TTBMs mais adequadas. Com isso, o processo de seleção se torna mais ágil, auxiliando a decisão final a respeito de quais TTBMs adotar em um projeto de software.

- **Quais melhorias você sugere para a caracterização do projeto de software?**

O participante 2 sugeriu a possibilidade de configurar a infra-estrutura para indicar quais atributos seriam aplicados primeiramente para se calcular a adequação, criando algo como um filtro sequencial de TTBMs por atributos. Atualmente a infra-estrutura só possui 1 nível para filtro de TTBMs.

- **O grau de adequação fornecido por *Porantim* e o gráfico de radar provido pela infra-estrutura foram úteis durante a seleção de TTBMs?**

O participante 1 indicou que o gráfico auxilia no processo de seleção, pois permite saber quais atributos estão sendo cobertos pelas TTBMs, porém há muita informação disponível, o que pode dificultar na seleção. O participante 2 sugeriu prover uma funcionalidade para sobrepor gráficos de radar de diferentes TTBMs, algo não provido atualmente. Apesar de tal sugestão, o participante elogiou o recurso.

- **Os indicadores de análise de combinação de TTBMs e os gráficos de Gauge provido pela infra-estrutura foram úteis durante a decisão de quais TTBMs combinar para um projeto de software?**

Ambos os participantes elogiaram tal recurso.

- **Qual o grau de dificuldade em usar a infra-estrutura computacional?**

Por fim, em relação à avaliação do uso da infra-estrutura, ambos os participantes indicaram que o grau de dificuldade em relação ao uso foi bastante baixo em relação ao apoio provido durante o processo de seleção.

A infra-estrutura computacional desenvolvida neste trabalho foi ainda aplicada com estudantes de pós-graduação em engenharia de software na COPPE/UFRJ no contexto de uma disciplina cujo objetivo era seguir as diferentes etapas para o desenvolvimento de um sistema ubíquo real. Nesse contexto, uma das etapas realizadas foi a seleção de TTBMs, além do planejamento dos testes para tal projeto. O objetivo não foi realizar um estudo de avaliação, mas apenas observar a viabilidade do uso da infra-estrutura no contexto de um projeto real.

Ao final, os estudantes preencheram um questionário indicando qual a impressão que tiveram com o uso da infra-estrutura. Os resultados indicaram que os participantes avaliaram de forma positiva as funcionalidades providas por Maraká para apoiar a seleção de TTBM utilizando a abordagem *Porantim*.

6.7 Considerações Finais do Capítulo

Neste Capítulo, foi apresentada a extensão da infra-estrutura computacional Maraká através da criação de um repositório de TTBMs e um componente para

aplicação da abordagem de apoio à seleção de TTBMs proposta neste trabalho, *Porantim*.

Dentre as suas principais funcionalidades de apoio à abordagem *Porantim*, esta infra-estrutura provê mecanismos de apoio para o cálculo automatizado dos graus de adequação entre um projeto de software previamente caracterizado e TTBMs disponíveis num repositório, além do cálculo automatizado dos indicadores providos por *Porantim* para apoiar na análise da combinação de TTBMs para projetos de software. Além disso, ela provê uma representação visual do grau de adequação de TTBMs utilizando Gráfico de Radar e dos indicadores de apoio à análise da combinação de TTBMs utilizando gráficos de Gauge.

O próximo Capítulo apresenta as considerações finais deste trabalho, descrevendo suas conclusões, resultados obtidos, limitações e futuras linhas de pesquisas a serem seguidas para continuidade desta pesquisa.

CAPÍTULO 7 - CONCLUSÕES

Neste capítulo são apresentadas as considerações finais sobre o trabalho realizado, incluindo os resultados obtidos, as limitações da pesquisa e os próximos passos a serem realizados com a continuidade desta pesquisa.

7.1 Considerações Finais

Teste Baseado em Modelos (TBM), assim como a aplicação das Técnicas de Teste Baseado em Modelos (TTBMs), tem sido campo de grande interesse da comunidade de Engenharia de Software ao longo dos anos. Há uma justificativa para isso, pois um dos elementos que podem influenciar na qualidade de um produto de software são os testes aplicados a ele, e nesse contexto, como discutido ao longo deste trabalho, TBM se apresenta como uma estratégia eficiente para a condução de testes de software.

A grande quantidade de TTBMs disponíveis com diferentes características, ausência de conhecimento técnico sobre elas e o fato das informações sobre as técnicas estarem dispersas em diferentes fontes tornam uma tarefa complexa para engenheiros de software a avaliação de quais TTBMs, quando combinadas, melhor se adéquam às características de um projeto de software.

Este trabalho propõe uma abordagem, chamada de *Porantim* e desenvolvida a partir de uma metodologia científica baseada na condução de estudos experimentais, que apóia a seleção combinada de TTBMs para projetos de software. Esse apoio é provido a partir da análise das principais características técnicas que descrevem as TTBMs e provendo conhecimento que auxilie na tomada de decisão a respeito de quais TTBMs são mais adequadas às características e requisitos de um projeto de software.

7.2 Resultados Obtidos

Analisando as sub-questões de pesquisa formuladas na Seção 1.3 deste documento, os seguintes resultados foram obtidos:

Questão 1) Seria possível melhorar o processo de seleção de TTBMs para projetos e organizações de software reduzindo o esforço, custo na utilização de uma ou mais

técnica de forma combinada em um projeto e aumentando-se a cobertura dos requisitos de teste definidos em um projeto de software?

RESULTADO: como mecanismo para se obter tais melhorias no processo de seleção de TTBMs, foi desenvolvida uma abordagem chamada *Porantim*, a partir de resultados de estudos secundários e primários, que visa prover indicadores que apóiam a análise da adequação de TTBMs e do impacto da combinação de TTBMs para projetos de software, avaliando dentre outras características a cobertura dos requisitos de teste definidos para o projeto. Estudos que avaliaram *Porantim* indicam uma redução significativa do tempo gasto para a seleção de TTBMs para projetos de software, o que estaria diretamente ligado a uma possível redução no esforço gasto para a realização desta tarefa em projetos de software. Foi desenvolvida ainda uma infra-estrutura computacional que apóia o uso da abordagem *Porantim*, provendo ainda melhorias no tempo e esforço no processo de seleção de TTBMs comparando-se com uma seleção sem apoio computacional.

Questão 2) O uso de uma abordagem de apoio à seleção de TTBMs mais adequadas dadas as características de um projeto de software e que indique o impacto da combinação de tais técnicas em variáveis do processo de teste auxiliaria na melhoria da efetividade, eficiência, completude, usabilidade e satisfação do usuário durante o processo de seleção de TTBMs para projetos de software?

RESULTADOS: *Porantim* foi avaliada em tais aspectos em relação à outra abordagem de seleção existente na literatura técnica. Os resultados do estudo indicaram que *Porantim* provê melhorias no que diz respeito à efetividade, eficiência e satisfação do usuário, e não apresentou diferença significativa no que diz respeito à completude e usabilidade do processo de seleção de TTBMs para projetos de software, conforme descrito no Capítulo 5.

7.3 Contribuições da Pesquisa

As principais contribuições desta pesquisa estão classificadas em 4 categorias:

- **Corpo de Conhecimento de TTBM's**

Um corpo de conhecimento pôde ser construído contendo a caracterização de 219 TTBMs identificadas na literatura técnica a partir de uma revisão sistemática. Este corpo de conhecimento foi estruturado a partir da consulta a especialistas em TBM por meio de um *survey* (pesquisa de opinião), onde os especialistas indicaram qual o

conjunto de atributos que possibilita uma caracterização de TTBMs e qual a relevância de cada atributo para a seleção de uma TTBM em um projeto de software.

Este corpo de conhecimento com as características das TTBMs identificadas está provido no banco de dados da infra-estrutura Maraká e também em um banco de dados da ferramenta JabRef. Ambos estão disponibilizados como resultado desta pesquisa.

- **Abordagem de Apoio à Seleção de TTBM: *Porantim***

Visando complementar a tarefa de selecionar TTBMs para projeto de software, foi definido ainda um processo de apoio responsável por coletar informações sobre o projeto de software no qual TTBMs devem ser aplicadas e prover conhecimento técnico a respeito da adequação e impacto de um conjunto de TTBMs em relação a um projeto de software.

A abordagem de seleção proposta foi avaliada. Um estudo experimental foi conduzido com dois grupos com diferentes características, estudantes de graduação com pouco conhecimento em engenharia e teste de software, e estudantes de pós-graduação com uma maior experiência prática em engenharia e teste de software. O objetivo deste estudo foi avaliar a completude, efetividade, eficiência, usabilidade e satisfação do usuário em relação ao uso de *Porantim* quando comparada a outra abordagem de apoio à seleção de técnicas de teste. Os resultados sugerem que *Porantim* seria mais eficiente e efetiva que a segunda abordagem no processo de seleção de TTBMs, e que para os demais aspectos possui comportamento similar.

- **Infra-estrutura Computacional de Apoio ao uso de *Porantim***

Foi construída uma infra-estrutura computacional para apoiar nas atividades que compõem o processo de seleção de TTBMs provido pela abordagem *Porantim*, visto que várias etapas são repetitivas e passíveis de erro quando realizadas manualmente. A infra-estrutura implementa o corpo de conhecimento de TTBMs e o processo de seleção de TTBMs proposto pela abordagem *Porantim*, além de prover outras facilidades no que diz respeito à manutenção e atualização do corpo de conhecimento de TTBMs. Isso permite que o corpo de conhecimento de TTBMs possa ser instanciado de acordo com as características específicas de uma organização de software.

Esta infra-estrutura foi utilizada por engenheiros de software de uma organização internacional que aplica TBM em seus projetos e estudantes de engenharia de software a fim de observar a viabilidade de seu uso. Os resultados, embora não conclusivos, indicam uma redução no tempo total gasto para seleção de

TTBMs, além de uma avaliação positiva do apoio provido pela infra-estrutura computacional pelos profissionais que a utilizam.

- **Metodologia Científica que apoia na Concepção de Tecnologias de Software a partir da condução de Estudos Secundários e Primários**

Durante o desenvolvimento desta pesquisa foi adotada uma metodologia científica baseada na condução de estudos secundários e primários para apoiar na concepção de novas tecnologias de software, ou seja, na fase onde uma tecnologia é idealizada e construída, antes de ser avaliada atraves de novos estudos. Esta metodologia busca agrupar resultados providos por revisões sistemáticas da literatura e a opiniões de especialistas em um tópico de pesquisa para a construção de um corpo de conhecimento sobre diferentes áreas de domínio.

Esta metodologia está descrita no Capítulo 1 deste trabalho e tem sido seguida por outras pesquisas científicas desenvolvidas por membros do Grupo de Engenharia de Software Experimental da COPPE/UFRJ.

7.4 Limitações

As limitações deste trabalho estão relacionadas, principalmente, a três itens: (1) escopo de avaliação da adequação de uma TTBM a um projeto de software, (2) escopo de avaliação do impacto da combinação de TTBMs no processo de testes e (3) avaliações da abordagem *Porantim* e da infra-estrutura computacional provida.

(1) Escopo de avaliação de uma TTBM a um projeto de software

Conforme citado no Capítulo 1 - Introdução, diversos aspectos podem influenciar na seleção de uma tecnologia para um projeto de software, tais como conhecimento técnico e habilidade da equipe de teste sobre tais tecnologias e sobre o domínio do projeto, cronograma do projeto, esforço e custo associado ao uso de uma tecnologia, aspectos políticos dentro de uma organização de software, dentre outros. No entanto, este trabalho foca apenas em aspectos técnicos como mecanismo para caracterização de uma TTBM e avaliação de sua adequação a um projeto de software, sem considerar os demais aspectos, que também possuem relevância na tomada de decisão a respeito da seleção de TTBMs para um projeto de software. Por exemplo, aspectos como o valor monetário para aquisição de uma ferramenta ou alguma decisão política de origem não técnica não são considerados pela abordagem proposta, o que reforça a afirmação de que o objetivo da abordagem proposta é prover conhecimento técnico que auxilie na tomada de decisão, mas não tomar a decisão pelo engenheiro de software ou substituir o seu papel nesta tarefa.

(2) Escopo de avaliação do impacto da combinação de TTBMs no processo de testes

Diversas variáveis de um processo de teste podem ser impactadas pelas TTBMs selecionadas para um projeto de software. No entanto, *Porantim* atualmente é direcionada à avaliação e estimativa do impacto de apenas três variáveis (Cobertura dos Requisitos de Teste do Projeto de Software, Esforço de Modelagem Salvo para a Criação dos Testes e Grau de Preparação da Equipe de Teste), apresentadas na Seção 4.2.4. Acredita-se que outras variáveis (ex: Custo para aquisição das ferramentas de apoio, Grau de complexidade/automação em relação ao uso das TTBMs selecionadas, Cobertura dos casos de teste a partir dos critérios de geração dos testes providos pelas TTBMs selecionadas) poderiam ser adotadas para avaliação do impacto de um conjunto de TTBMs em um projeto de software e resolver possíveis conflitos de características entre elas.

(3) Avaliações da abordagem *Porantim* e da infra-estrutura computacional provida

A abordagem foi avaliada em relação a outra abordagem de apoio à seleção de técnicas de teste, e inclusive apresentou resultados positivos em relação à segunda abordagem. No entanto, novos estudos podem ser realizados, inclusive com outras abordagens que possam ser aplicadas à seleção de TTBMs a fim de se obter mais resultados que auxiliem no amadurecimento desta nova tecnologia que está sendo proposta neste trabalho.

Em relação à infra-estrutura computacional desenvolvida, esta foi aplicada por engenheiros de software que utilizam TTBMs no seu dia-a-dia e por estudantes de pós-graduação em engenharia de software. No entanto, apesar da avaliação ter sido realizada com dados de projetos e TTBMs reais, esta não foi usada no contexto de um projeto real ao longo de um processo de desenvolvimento de software, mas sim em um ambiente isolado para avaliação da infra-estrutura computacional. Dessa forma, torna-se importante realizar uma avaliação da infra-estrutura computacional desenvolvida no contexto de um processo de desenvolvimento de um projeto real.

7.5 Futuras Linhas de Pesquisa

Há a intenção de continuar a pesquisa apresentada neste trabalho seguindo os seguintes direcionamentos:

- Conforme sugerido por um participante do *Workshop on Automated Software Test* (AST) em 2009, *Porantim* poderia prover um apoio não mais apenas à seleção de TTBMs para um projeto de software, mas sim para um portfólio de projetos de software, ou seja, deseja-se saber quais são as TTBMs mais adequadas às características de um grupo de projetos de software. Sendo assim, a relação entre projeto de software e TTBM deixaria de ser de 1:N, como é atualmente em *Porantim*, e passaria a ser de M:N. O objetivo a ser seguido nesta linha de pesquisa é prover uma extensão de *Porantim* que possibilite a seleção de TTBMs a partir das características de um portfólio de projetos de software.
- Seguindo outra sugestão apresentada no mesmo fórum, poderia ser provida uma abordagem que não apenas apóie a seleção de TTBMs a partir de características e requisitos de um projeto de software, mas inverter o processo de tomada decisão. Sendo assim, a partir das características de um conjunto de TTBMs que se deseja aplicar em um projeto de software, identificar quais seriam as características e requisitos mais adequados para o projeto, como por exemplo, qual linguagem de modelagem adotar no projeto de software a ser desenvolvido ou qual linguagem de programação adotar. O objetivo a ser seguido nesta linha de pesquisa é prover uma nova abordagem de apoio à tomada de decisão a partir de características de TTBMs, sendo contextualizada ao conceito de Desenvolvimento Dirigido a Teste (ou TDD, do inglês *Test Driven Development*) [JANZEN e SAIEDIAN, 2005].
- Por fim, entende-se que a aplicação de TTBMs em projetos de software pode ser dividida em três momentos principais: a seleção das TTBMs, a sua utilização no projeto de software e a sua avaliação após o uso no projeto de software. Este trabalho está relacionado apenas ao primeiro momento (seleção de TTBMs), porém deseja-se prover apoio aos outros dois momentos citados. Sendo assim, o objetivo a ser seguido nesta linha de pesquisa é prover uma abordagem que apóie na utilização de TTBMs em projetos de software, através da coleta de informações sobre o uso de tais técnicas e do apoio às tomadas de decisões que ocorrem durante o uso de uma TTBM, como por exemplo a seleção de qual critério de geração dos testes adotar em cada parte do projeto, e à avaliação de TTBMs como uma forma de prover conhecimento sobre o uso de tais TTBMs em projetos reais, que consiste em uma das principais carências na área de Teste Baseado em Modelos.

REFERÊNCIAS BIBLIOGRÁFICAS

ABDURAZIK, A. OFFUTT, J. (2000), “Using UML Collaboration Diagrams for Static Checking and Test Generation”, Third International Conference The Unified Modeling Language (UML’2000), York, UK, Outubro, pp. 383-395.

ALAGAR, V.; ORMANDJIEVA, O.; ZHENG, M. (2000), ‘Specification-based testing for real-time reactive systems”, Proceedings. 34th International Conference on Technology of Object-Oriented Languages and Systems (TOOLS), pp. 25-36.

ASADA, T., SWONGER, R. F., BOUNDS, N. et al. (1992), “The Quantified Design Space: A Tool for the Quantitative Analysis of Design”, SEI Technical Report CMU/SEI-92-TR-213, Carnegie Mellon University.

ARANDA, G. N., VIZCAINO, A., CECHICH, A., PIATTINI, M. (2006), “Technology Selection to Improve Global Collaboration”, In: International Conference on Global Software Engineering (ICGSE), Outubro, pp. 223-232.

BASILI, V.R., CALDIERA, G., ROMBACH, H.D. (1994), “Goal Question Metric Approach,” Encyclopedia of Software Engineering, John Wiley & Sons, Inc, pp. 528-532.

BASILI, V. R., ROMBACH, H. D. (1991), “Support for comprehensive reuse”. Software Engineering Journal 6(5): September, pp. 303-316.

BEIZER, B. (1990), Software testing techniques, 2nd ed., Van Nostrand Reinhold Co., New York, NY.

BERGER, P. (2003), “Instanciação de Processos de Software em Ambientes Configurados na Estação TABA**”**, Dissertação de Mestrado COPPE/UFRJ. Rio de Janeiro.

BERTOLINO, A. (2004), “Guide to the Knowledge Area of Software Testing. Software Engineering Body of Knowledge”, IEEE Computer Society, Fevereiro, http://www.swebok.org .

BIOLCHINI, J.; MIAN, P.G.; NATALI, A.C.; TRAVASSOS, G.H. (2005), “Systematic Review in Software Engineering: Relevance and Utility”, Relatório Técnico ES-679/05, PESC-COPPE/UFRJ. Disponível em http://www.cos.ufrj.br.

BIRK, A. (1997), “Modelling the application domains of software engineering technologies”, Proceedings of the 12th International Conference on Automated

Software Engineering (ASE), Lake Tahoe, CA, November, IEEE Computer Society, pp 291.

BOLDRINI, J. L., COSTA, S. R., FIGUEIREDO, V. L., et al. (1980), “Álgebra Linear”, 3 ed., capítulo 8, Harper & Row do Brasil.

BOUSQUET, L. d., ZUANON, N. (1999), “An Overview of Lutess: A Specification-Based Tool for Testing Synchronous Software”. Proceedings of the 14th IEEE international Conference on Automated Software Engineering (ASE), p. 208.

BRIAND, L. C., LABICHE, Y. (2001), “A UML-Based Approach to System Testing”, Proceedings of the 4th international Conference on the Unified Modeling Language, Modeling Languages, Concepts, and Tools, vol. 2185, Londres, Outubro, pp. 194-208.

CHANG, J.; RICHARDSON, D. J.; SANKAR, S. (1996), “Structural specification-based testing with ADL”, Proceedings of the 1996 International Symposium on Software Testing and Analysis ISSTA, , ACM Press, p. 21.

CHOW, T.S. (1978), “Testing software design modeled by finite-state machines”, IEEE Transaction Software Engineering, vol. SE-4, pp. 178-187, Março.

CHUNG, I. S., KIM, H. S., BAE, H. S., KWON, Y. R., LEE, B. S. (1999), “Testing of Concurrent Programs Based on Message Sequence Charts”, Proceedings of the international Symposium on Software Engineering For Parallel and Distributed Systems (PDSE), pp. 72.

DALAL, S.; JAIN, A.; KARUNANITHI, N.; LEATON, J.; LOTT, C.; PATTON, G.; HOROWITZ, B. (1999), “Model-based testing in practice”, Proceedings of the 1999 International Conference on Software Engineering (ICSE’99), Maio, pp. 285-294.

DIAS NETO, A. C., (2006) "Uma Infra-estrutura Computacional para Apoiar o Planejamento e Controle de Testes de Software", Dissertação de M.Sc., COPPE/UFRJ, Abril.

DIAS-NETO, A. C.; TRAVASSOS, G. H. (2006), “Maraká: Uma Infra-estrutura Computacional para Apoiar o Planejamento e Controle de Testes de Software”. In: Simpósio Brasileiro de Qualidade de Software, Vila Velha-ES, Junho.

DIAS-NETO, A.C.; SUBRAMANYAN, R.; VIEIRA, M.; TRAVASSOS, G.H. (2007a), “Characterization of Model-based Software Testing Approaches”, Relatório Técnico ES-713/07, PESC-COPPE/UFRJ. Disponível em http://www.cos.ufrj.br/uploadfiles/1188491168.pdf.

DIAS-NETO, A. C., SUBRAMANYAN, R.; VIEIRA, M.; TRAVASSOS, G.H. (2007b), “A survey on model-based testing approaches: a systematic review”. Proceedings of the 1st ACM international Workshop on Empirical Assessment of Software Engineering Languages and Technologies (WEASELTech'07): Held in Conjunction with the 22nd IEEE/ACM international Conference on Automated Software Engineering (ASE) 2007 (Atlanta, Georgia, Novembro), pp 31-36. DOI= http://doi.acm.org/10.1145/1353673.1353681

DIAS-NETO, A.C., SPÍNOLA, R.O., BOTT, A., TRAVASSOS, G.H. (2007c), “Estratégia de Teste de Software no Desenvolvimento Incremental de um Sistema de Informação”, No: 1st Brazilian Workshop on Systematic and Automated Software Testing (SAST), João Pessoa-PB, Outubro.

DIAS-NETO, A.C.; SUBRAMANYAN, R.; VIEIRA, M.; TRAVASSOS, G.H.; FORREST, S. (2008), “Improving Evidence about Software Technologies: A Look at Model-Based Testing”, IEEE Software, Vol. 25, Issues 3, pp 10-13, Maio.

DIAS-NETO, A. C., TRAVASSOS, G. H. (2008a), “Supporting the selection of model-based testing approaches for software projects”. Proceedings of the 3rd international Workshop on Automation of Software Test (Leipzig, Germany, May). AST '08. pp. 21-24, Maio, DOI= http://doi.acm.org/10.1145/1370042.1370047

DIAS-NETO, A.C., TRAVASSOS, G.H. (2008b), “Surveying on Model Based Testing Approaches Characterization Attributes”, Proceeding of International Symposium on Empirical Software Engineering and Measurement (ESEM’08), Short Paper, Outubro, Kaiserslautern, Alemanha.

DIAS NETO, A.C.; TRAVASSOS, G.H.; (2008c), “Estratégia para Apoiar a Seleção de Abordagens de Teste Baseado em Modelos para Projetos de Software”, No: VII Simpósio Brasileiro de Qualidade e Software (SBQS’2008), Florianópolis, SC, Junho.

DIAS NETO, A.C., TRAVASSOS, G.H. (2008d), “Uma Estratégia de Apoio à Seleção de Abordagens de Teste Baseado em Modelos para Projetos de Software”, No: Workshop de Teses e Dissertações em Qualidade de Software (WTDQS’08), Junho, Florianópolis, SC.

DIAS-NETO, A.C.; TRAVASSOS, G.H. (2009a), “*Porantim*: An Approach to Support the Combination and Selection of Model-Based Testing Techniques”, In: 4th Workshop on Automation of Software Test, Vancouver, Maio.

DIAS-NETO, A.C.; TRAVASSOS, G.H. (2009b), “Model-based Testing Approaches Selection for Software Projects”, In: Information and Software Technology (AST’08 special edition), Julho, DOI: 10.1016/j.infsof.2009.06.010.

DIAS-NETO, A.C., TRAVASSOS, G.H. (2009c), “Evaluation of {model-based} Testing Techniques Selection Approaches: an External Replication”, Proceeding of International Symposium on Empirical Software Engineering and Measurement (ESEM’09), Outubro, Lake Buena Vista, EUA.

FRIEDMAN, G.; HARTMAN, A.; NAGIN, K.; SHIRAN, T. (2002), “Projected state machine coverage for software testing”, Proceedings of the 2002 ACM SIGSOFT international symposium on Software testing and analysis (ISSTA '02), ACM Press, p. 27.

HARTMANN, A., NAGIN, K. (2004), “The AGEDIS tools for model based testing”. SIGSOFT Software Engeneering Notes 29, 4 (Julho), pp. 129-132. DOI= http://doi.acm.org/10.1145/1013886.1007529.

HAMBURG, Morris (1980), “Basic Statistics: A Modern Approach”, Journal of the Royal Statistical Society, Series A (General), Vol. 143, No. 1, 2ª edição.

IEEE Standard 610-1990: IEEE Standard Glossary of Software Engineering Terminology, IEEE Press.

JACCARD, Paul (1901), “Étude comparative de la distribution florale dans une portion des Alpes et des Jura”, Bulletin del la Société Vaudoise des Sciences Naturelles 37, pp. 547-579.

JANZEN, D.; SAIEDIAN, H. (2005), “Test-Driven Development: Concepts, Taxonomy, and Future Direction”. Computer 38, Vol. 9 (Setembro), pp. 43-50. DOI= http://dx.doi.org/10.1109/MC.2005.314

JURISTO, N.; MORENO, A.M.; VEGAS, S. (2004), “Reviewing 25 years of testing Technique experiments”. Empirical Software Engineering: An International Journal, 9(1), p. 7-44, Março.

KALINOWSKI, M., SPÍNOLA, R. O., DIAS-NETO, A. C., BOTT, A., TRAVASSOS, G. H. (2007), “Inspeções de Requisitos de Software em Desenvolvimento Incremental: Uma Experiência Prática”. No: Simpósio Brasileiro de Qualidade de Software, Porto de Galinhas, Junho.

KANER, C. (2006), “Exploratory Testing”, Florida Institute of Technology, Quality Assurance Institute Worldwide Annual Software Testing Conference, Orlando, FL, Novembro.

KIM, Y.; HONG, H.; BAE, D.; CHA, S. (1999), “Test cases generation from UML state diagrams”, IEE Proceedings, Vol. 146, pp. 187-192.

KONTIO, J. (1995), OTSO: A Systematic Process for Reusable Software Component Selection, Computer Science Technical Report CS-TR-3478, University of Maryland.

MAIDEN, N. A. M.; RUGG, G. (1996), “ACRE: Selecting methods for requirements acquisition”, Software Engineering Journal 11(3), pp. 183-192.

MENDONÇA. M. G.. MALDONADO. J. C.. OLIVEIRA. M. C.. CARVER. J.. FABBRI. S. C.. SHULL. F.. TRAVASSOS. G. H.. HÖHN. E. N.. and BASILI. V. R. (2008). “A Framework for Software Engineering Experimental Replications”. In Proceedings of the 13th ICECCS. Vol. 00. Washington. DC. pp. 203-212.

MENZIES, T., OWEN, D., CUKIC, B. (2002), “Saturation Effects in Testing of Formal Models”. In: 13th international Symposium on Software Reliability Engineering (Issre'02), Washington, DC, p. 15.

MONETA, C., VERNAZZA, G., ZUNINO, R. (1990), “A Vectorial Definition of Conceptual Distance for Prototype Acquisition and Refinement”. Technical Report TUM-I9019, Technical University Munich.

PARISSIS, I., VASSEY, J. (2003), “Thoroughness of Specification-Based Testing of Synchronous Programs”. Proceedings of the 14th international Symposium on Software Reliability Engineering (ISSRE), Washington, DC, p. 191.

PRASANNA, M.; SIVANANDAM, S.N.; VENKATESAN, R.; SUNDARRAJAN, R. (2005), “Survey on Automatic Test Case Generation”, Academic Open Internet Journal, Vol. 15, disponível em http://www.acadjournal.com/2005/v15/part6/p4/.

PRETSCHNER, A. (2005), “Model-based testing”, Proceedings of 27th International Conference on Software Engineering, (ICSE‘05), pp. 722-723.

RAMAMOORTHY, C. V.; HO, S. F. & CHEN, W. T. (1976), “On the automated generation of program test data”, IEEE Transactions on Software Engineering, SE-2(4):293–300, Dezembro.

RAPPS, S., WEYUKER, E.J., (1982), “Data Flow analysis techniques for test data selection”, In: International Conference on Software Engineering, p. 272-278, Tokio, Setembro.

ROCHA, A. R. C., MALDONADO, J. C., WEBER, K. C. (2001), “Qualidade de software – Teoria e prática”, Prentice Hall, São Paulo.

SANTHANAM, P.; CHILLAREGE, R. (1998), “Technical Converence on Strategic Technologies”, Presented at the 1998 Korea-U.S. Tysons Corner, VA., October 22-24.

SPINOLA, R. O. ; DIAS-NETO, A. C.; TRAVASSOS, G. H. (2008), “Abordagem para Desenvolver Tecnologia de Software com Apoio de Estudos Secundários e Primários“. In: Experimental Software Engineering Latin American Workshop (ESELAW), Salvador, Novembro.

STOBIE, K. (2005), “Model Based Testing in Practice at Microsoft”, In: Electronic Notes in Theoretical Computer Science, vol. 111, pp. 5 – 12.

UTTING, M., PRETSCHNER, A., LEGEARD, B., “A taxonomy of model-based testing”, Technical report 04/2006, Department of Computer Science, University of Waikato, Abril, 2006.

UTTING, M.; LEGEARD, B.; (2007), “Practical Model-Based Testing: A Tools Approach”, ISBN-13: 978-0-12-372501-1, Morgan-Kaufmann.

VEGAS, S.; BASILI, V. (2005), “A Characterization Schema for Software Testing Techniques”, Empirical Software Engineering, v.10 n.4, p.437-466, Outubro.

VIEIRA, M.; LEDUC, J.; HASLING, B.; SUBRAMANYAN, R.; KAZMEIER, J. (2006), “Automation of GUI testing using a model-driven approach”, Proceedings of the 2006 International Workshop on Automation of Software Test (AST'06), ACM Press.

WOJCICKI, M. A.; STROOPER, P. (2007), “An Iterative Empirical Strategy for the Systematic Selection of a Combination of Verification and Validation Technologies”. Proceedings of the 5th international Workshop on Software Quality (May 20 - 26). International Conference on Software Engineering. DOI= http://dx.doi.org/10.1109/WOSQ.2007.4

XAVIER, J. R.; WERNER, C.M.L.; TRAVASSOS, G.H.; (2002), “Uma Abordagem para a Seleção de Padrões Arquiteturais Baseada em Características de Qualidade”, XVI Simpósio Brasileiro de Engenharia de Software, Gramado, RS, Brasil.

APÊNDICE A – FONTES E ARTIGOS UTILIZADOS NA REVISÃO SISTEMÁTICA SOBRE TÉCNICAS DE TESTE BASEADO EM MODELOS

Este apêndice apresenta a lista de fontes de artigos (bibliotecas digitais, sites, conferências) e a lista de artigos utilizados na revisão sistemática sobre técnicas de teste baseado em modelos descrita na Seção 3.2.

AP A.1 Métodos de Seleção das Fontes de Artigos

As fontes de artigos foram acessadas através da Internet. No contexto desta revisão sistemática, não foi aplicada busca manual.

AP A.2. Lista de Fontes

- ACM Digital Library: http://portal.acm.org
- EI Compendex: http://www.engineeringvillage.com
- IEEEXplorer: http://ieeexplore.ieee.org
- INSPEC: http://web5.silverplatter.com/webspirs/
- SCOPUS: http://www.scopus.com/search/form.url
- Web of Science: http://www.isiwebofknowledge.com
- The Software Quality Engineering Laboratory (SQUALL): Technical Reports (Carleton University): http://squall.sce.carleton.ca/pubs_tech_rep.html. Will be used the following technical reports:
    - A State-based Approach to Integration Testing for Object-Oriented Programs.
    - A UML-Based Approach to System Testing.
    - Improving State-Based Coverage Criteria Using Data Flow Information.
    - Revisiting Strategies for Ordering Class Integration Testing in the Presence of Dependency Cycles.
    - Towards Automated Support for Deriving Test Data from UML Statecharts.
- Online Papers About Model-Based Testing available at: http://www.geocities.com/model_based_testing/online_papers.htm (September 18, 2006). Will be used the following papers:
    - ABDURAZIK, A.; OFFUTT, J.; “Using UML Collaboration Diagrams for Static Checking and Test Generation”; UML 2000 - The Unified Modeling Language.

Advancing the Standard. Third International Conference, York, UK, October 2000, Proceedings, Springer, 2000, 1939, 383-3.

- ANDREWS, A.; OFFUTT, J.; ALEXANDER., R.; “Testing web applications by modeling with FSMs”; Software Systems and Modeling, 2005, 4(2).
- MEYER, Steve; SANDFOSS, Ray; “Applying Use-Case Methodology to SRE and System Testing”; STAR West Conference, Oct. 1998, pp. 1-16.

- Kamran Ghani’s Website (PhD Student at York University). Some papers related to Model-Based Testing available at: http://www-users.cs.york.ac.uk/~kamran/papers.htm#R2 (September 18, 2006). Will be used the following papers:
  - SCHEETZ, M.; von MAYHAUSER, A.; FRANCE, R.; DAHLMAN, E.; HOWE, A.E.; “Generating Test Cases from an OO Model with an AI Planning System”, ISSRE '99: Proceedings of the 10th International Symposium on Software Reliability Engineering, IEEE Computer Society, 1999, 250.
  - OFFUTT, Jeff; LIU, Shaoying; ABDURAZIK, Aynur, AMMANN, Paul; “Generating Test Data from State-Based Specifications”, Journal of Software Testing, Verification and Reliability, John Wiley & Sons, Ltd, No.13, 2003, pp. 25-53.
  - CRICHTON, Charles; CAVARRA, Alessandra; DAVIES, Jim; “Using UML for Automatic Test Generation” Proceedings of Automated Software Engineering (ASE), 2001.
- Informatik 2006 - http://www.informatik2006.de/272.html
  - NEBUT, C., FLEUREY, F.; “Automatic Test Generation: A Use Case Driven Approach”, IEEE Trans. Software Engineering, IEEE Press, 2006, 32, 140-155 (quoted by one paper).
  - BERNARD, E., BOUQUET, F., CHARBONNIER, A., LEGEARD, B., PEUREUX, F., UTTING, M., TORREBORRE, E.; “Model-Based Testing from UML Models”, In proceedings: Informatik, 2006 (printed copy).
  - ERNITS, J., KULL, A., RAINED, K., VAIN, J.; “Generating Test Sequences from UML Sequence Diagrams and State Diagrams”, In proceeding: Informatik, 2006 (printed copy).

AP A.3. Lista de Artigos Identificados na Revisão Sistemática sobre TTBMs

AP A.3.1 Primeira Execução – Agosto/2006

Categoria	Título	Autores	Fonte	Ano
D	A case for test-code generation in model-driven systems	Rutherford and Wolf	GPCE	2003
B	A Choice Relation Framework for Supporting Category-Partition Test Case Generation	Chen *et al*.	IEEE Transactions on Software Engineering	2003
E	A flexible environment to evaluate state-based test techniques	Hierons	ACM SIGSOFT Software Engineering Notes	2004
B	A formal approach to requirements based testing in open systems standards	Leathrum and Liburdy	International Conference on Requirements Engineering	1996
E	A framework and tool support for the systematic testing of model-based specifications	Miller and Strooper	TOSEM	2003
D	A framework for specification-based class testing	Liu *et al*.	ICECCS	2002
D	A Framework for Specification-Based Testing	Stocks and Carrington	IEEE Transactions on Software Engineering	1996
NC[18]	A framework for table driven testing of Java classes	Daley *et al*.	Software—Practice & Experience	2002
D	A Generic Model-Based Test Case Generator	Popovic and Velikic	ECBS	2005
B	A holistic approach to test-driven model checking	Belli and Guldali	International conference on Innovations in Applied Artificial Intelligence	2005
E	A hybrid component-based system development process	Teiniker *et al*.	31st EUROMICRO Conference on Software Engineering and Advanced Applications	2005
A	A method for the automatic generation of test suites from object models	Cavarra *et al*.	ACM symposium on Applied computing	2003
B	A method of generating massive virtual clients and model-based performance test	Kim	QSIC	2005
A	A methodology and a framework for model-based testing	Lucio *et al*.	Lecture Notes in Computer Science	2005
NC	A Methodology of Verification and Testing of Large Software Systems	Lipaev	Programming and Computing Software	2003
E	A model-based statistical usage testing of communication protocols	Popovic *et al*.	International Symposium and Workshop on Engineering of Computer Based Systems	2006
B	A model-to-implementation mapping tool for automated model-based GUI testing	Paiva *et al*.	ICFEM	2005
B	A new approach to test case generation based on real-time process algebra (RTFA)	Yao and Wang	Canadian Conference on Electrical and Computer Engineering	2004
E	A practical approach to modified condition/decision coverage	Hayhurst and Veerhusen	DASC	2001
C	A Practical Approach to UML-based Derivation of Integration Tests	Basanieri and Bertolino	QWE	2000
E	A reliability estimator for model based software testing	Sayre and Poore	ISSRE	2002
E	A rigorous method for test templates generation from object-oriented specifications	Periyasamy and Alagar	Software Testing Verification and Reliability	2001
E	A schema language for coordinating construction and composition of partial behavior descriptions	Grieskamp and Kicillof	SCESM	2006
E	A specification driven hierarchical test methodology	Sathianathan and Smith	IEEE International ASIC Conference and Exhibit	1993

[18] NC = Não Classificado.

D	A specification-based adaptive test case generation strategy for open operating system standards	Watanabe and Sakamura	ICSE	1996
NC	A specification-based case study from test class framework	Liu and Miao	Journal of Shanghai University	2001
C	A State-based Approach to Integration Testing for Object-Oriented Programs	Ali *et al*.	Technical Report of Carleton University	2005
NC	A study of user acceptance tests	Leung and Wong	Software Quality Control	1997
NC	A test data generation tool based on inter-relation of fields in the menu structure	Lee and Choi	Journal of KISS: Computing Practices	2003
E	A Test Generation Strategy for Pairwise Testing	Tsuchiya and Kikuno	TOSEM	2002
B	A test sequence selection method for statecharts	Hong *et al*.	Software Testing Verification & Reliability	2000
D	A theory of specification-based testing for object-oriented software	Barbey *et al*.	EDCC	1996
E	A tool for testing synchronous software	Parissis	Achieving Quality in Software	1996
A	A transition-based strategy for object-oriented software testing	Traore	ACM symposium on Applied computing	2003
A	A UML-Based Approach to System Testing	Briand and Labiche	Technical Report of Carleton University	2002
NC	A use case driven testing process: towards a formal approach based on UML collaboration diagrams	Badri *et al*.	Formal Approaches to Software Testing	2004
E	Action machines - towards a framework for model composition, exploration and conformance testing based on symbolic computation	Grieskamp *et al*.	QSIC	2005
B	Action refinement in conformance testing	Van Der Bijl *et al*.	Lecture Notes in Computer Science	2005
E	Activity based SW process as basis for ISO 9000	Mahabala	CSI Communications	1993
D	Adding natural relationships to simulink models to improve automated model-based testing	Boden and Busser	AIAA/IEEE Digital Avionics Systems Conference	2004
D	ADLscope: an automated specification-based unit testing tool	Chang and Richardson	ASE	1998
NC	An adaptive use case design driven testing	Kim *et al*.	ISCA	2000
E	An analysis and testing method for Z documents	Ciancarini *et al*.	Annals of Software Engineering	1997
E	An analysis of rule coverage as a criterion in generating minimal test suites for grammar-based software	Hennessy and Power	ASE	2005
E	An Analysis of Test Data Selection Criteria Using the RELAY Model of Fault Detection	Richardson and Thompson	IEEE Transactions on Software Engineering	1993
B	An approach to detecting domain errors using formal specification-based testing	Chen and Liu	Asia-Pacific Software Engineering Conference	2004
NC	An approach to generate integration test cases based on UML collaboration diagrams	zhang, W.; dong, L. & liang, Z.	Acta Electronica Sinica	2004
D	An approach to integration testing based on data flow specifications	Chen *et al*.	Lecture Notes in Computer Science	2005
E	An approach to specification-based testing systems	Zin *et al*.	Software Quality Engineering	1997
E	An approach to verification and validation of a reliable multicasting protocol	Callahan and Montgomery	ISSTA	1996
NC	An experimental evaluation of a higher-ordered-typed-functional specification-based test-generation technique	Sinha and Smidts	Empirical Software Engineering	2006
D	An explorative journey from architectural tests definition down to code tests execution	Bertolino *et al*.	ICSE	2001
E	An extended fault class hierarchy for specification-based testing	Lau and Yu	ACM Transactions on Software Engineering and Methodology	2005
NC	An extended finite state machine based generation method of test suite	Guo and Ping	Journal of Software	2001
E	An improved model-based method to test circuit faults	Cheng *et al.*	Theoretical Computer Science	2005
E	An integrated method for designing user interfaces based on tests	Schilling *et al*.	A-MOST	2005
NC	An object-based data flow testing approach for Web applications	Liu *et al*.	International Journal of Software Engineering and Knowledge Engineering	2001
D	An overview of Lutess: A specification-based tool for testing synchronous software	du Bousquet and	ASE	1999

		Zuanon		
E	An overview of model-based testing	Utting	Technique et Science Informatiques	2006
D	Analyzing software architectures with Argus-I	Vieira *et al*.	ICSE	2000
E	Applicability of non-specification-based approaches to logic testing for software	Kobayashi *et al*.	International Conference on Dependable Systems and Networks	2001
NC	Application of software quality assurance methods in validation and maintenance of reactor analysis computer codes	Reznik	Reactor Physics and Reactor Computations	--
D	Applying conventional testing techniques for class testing	Chung *et al*.	IEEE Computer Society's International Computer Software & Applications Conference	1996
NC	Applying extended finite state machines in software testing of interactive systems	Fantinato and Jino	Interactive Systems: Design, Specification, and Verification	2003
E	Applying models in your testing process	Rosaria and Robinson	Information and Software Technology	2000
NC	Applying mutation analysis to SDL specifications	Kovacs *et al*.	SDL 2003: System Design, Proceedings	2003
A	Applying Use-Case Methodology to SRE and System Testing	Meyer and Sandfoss	STAR West Conference	1998
NC	Approach to specification-based testing systems	Mohd Zin *et al*.	SQE	1997
E	Auto-generating test sequences using model checkers: a case study	Heimdahl *et al*.	Formal Approaches to Software Testing	2003
NC	Automated boundary testing from Z and B	Legeard *et al*.	FME	2002
E	Automated consistency and completeness checking of testing models for interactive systems	Paradkar and Klinger	International Computer Software and Applications Conference	2004
A	Automated Generation of Statistical Test Cases from UML State Diagrams	Chevalley and Fosse	COMPSAC	2005
E	Automated generation of test programs from closed specifications of classes and test cases	Leow *et al*.	ICSE	2004
NC	Automated generation of test scenarios based on UML specification	Nagy	Automatizace	2005
E	Automated model-based testing of Chi simulation models with TorX	Van Osch	Lecture Notes in Computer Science	2005
D	Automated Test Case Generation for Programs Specified by Relational Algebra Queries	Tsai *et al*.	IEEE Transactions on Software Engineering	1990
NC	Automated test case generation from IFAD VDM++ specifications	Nadeem and Jaffar-Ur-Rehman	WSEAS Transactions on Computers	2005
B	Automated test oracles for GUIs	Memon *et al*.	ACM SIGSOFT international symposium on Foundations of software engineering	2000
B	Automated testing from object models	Poston	Communications of the ACM	1994
D	Automated Testing of Classes	Buy *et al.*	ISSTA	2000
C	Automated TTCN-3 test case generation by means of UML sequence diagrams and Markov chains	Beyer *et al*.	ATS	2003
B	Automated validation test generation	Weber *et al*.	DASC	1994
E	Automated verification and test case generation for input validation	Liu and Tan	AST	2006
C	Automated, contract-based user testing of commercial-off-the-shelf components	Briand *et al*.	ICSE	2006
A	Automated-generating test case using UML statechart diagrams	Kansomkeat and Rivepiboon	SAICSIT	2003
NC	Automatic construction of a class state-based testing model using method specifications	Al-Dallal and Sorenson	IASTED: International Conference on Computer Science and Technology	2003
D	Automatic extraction of abstract-object-state machines from unit-test executions	Xie *et al*.	ICSE	2006
E	Automatic generation of test cases from Boolean specifications using the MUMCUT strategy	Yu *et al*.	ASE	2006
E	Automatic Generation of Test Oracles—From Pilot Studies to Application	Feather and Smith	ASE	2001
A	Automatic test case generation for UML activity diagrams	Mingsong *et al*.	AST	2006
E	Automatic test generation for predicates	Paradkar *et al*.	ISSRE	1996

A	Automatic Test Generation: A Use Case Driven Approach	Nebut and Fleurey	IEEE Transaction Software Engineering	2006
NC	Automatic UML-based test data generating tool: AUTEG	Cheongah and Byoungju	Journal of KISS: Computing Practices	2002
D	Automatically testing interacting software components	Gallagher and Offutt	AST	2006
B	Automating formal specification-based testing	Donat	International Joint Conference CAAP/FASE on Theory and Practice of Software Development	1997
A	Automating impact analysis and regression test selection based on UML designs	Briand *et al*.	ICSM	2002
D	Automating software module testing for FAA certification	Santhanam	ACM SIGAda international conference on Ada	2001
E	Automating Specification-Based Software Testing	Poston	IEEE Computer Society Press	1997
NC	Automating test generation for discrete event oriented embedded systems	Cunning and Rozenblit	Journal of Intelligent and Robotic Systems	2005
A	Automation of GUI testing using a model-driven approach	Vieira *et al*.	AST	2006
NC	Axiomatic assessment of logic coverage software testing criteria	Liu and Miao	Ruan Jian Xue Bao/Journal of Software	2004
E	Behavior modeling technique based on EFSM for interoperability testing	Noh *et al*.	ICCSA	2005
NC	Behavior-based integration testing of software systems: a formal scenario approach	Pei *et al*.	International Conference on Systems Integration	1994
NC	Benefits of using model-based testing tools	Bruno *et al*.	Symposium on Software Quality	1995
D	Black-box testing using flowgraphs: An experimental assessment of effectiveness and automation potential	Edwards	Software Testing Verification and Reliability	2000
B	Boundary Coverage Criteria for Test Generation from Formal Models	Kosmatov *et al*.	ISSRE	2004
E	Building testable software	Zucconi and Reed	ACM SIGSOFT Software Engineering Notes	1996
E	Cause-effect graphing analysis and validation of requirements	Nursimulu and Probert	Conference of the Centre for Advanced Studies on Collaborative research	1995
D	Combining algebraic and model-based test case generation	Dan and Aichernig	Lecture Notes in Computer Science	2005
B	Combining behavior and data modeling in automated test case generation	Schroeder *et al*.	QSIC	2003
E	Comparison of fault classes in specification-based testing	Okun *et al*.	Information and Software Technology	2004
E	Confirming configurations in EFSM testing	Petrenko *et al*.	IEEE Transactions on Software Engineering	2004
B	Constructing multiple unique input/output sequences using metaheuristic optimisation techniques	Guo *et al*.	IEE Proceedings Software	2005
E	Constructing test suites for interaction testing	Cohen *et al*.	ICSE	2003
E	Continuous TTCN-3: testing of embedded control systems	Schieferdecker *et al*.	SEAS	2006
D	Controlling test case explosion in test generation from B formal models	Legeard *et al*.	Software Testing, Verification & Reliability'	2004
B	Coverage metrics for requirements-based testing	Whalen *et al*.	ISSTA	2006
D	Coverage-directed test generation with model checkers: challenges and opportunities	Devaraj *et al*.	COMPSAC	2005
E	Criteria for generating specification-based tests	Offutt *et al*.	ICECCS	1999
NC	DAS-BOOT: design-, architecture- and specification-based approaches to object-oriented testing	Richardson	ACM SIGSOFT Software Engineering	2000
B	Data abstraction and constraint solving for conformance testing	Calame *et al.*	APSEC	2005
E	Data flow testing as model checking	Hong *et al*.	ICSE	2003
NC	Data Generation for Path Testing	Mansour and Salame	Software Quality Control	2004
D	DeepTrans - a model-based approach to functional verification of address translation mechanisms	Adir *et al.*	4th International Workshop on Microprocessor Test and Verification	2003
E	Demonstration of an operational procedure for the model-based testing of CTI systems	Hagerer *et al*.	International Conference on Fundamental Approaches to Software Engineering	2002
B	Dependence analysis in reduction of requirement based test suites	Vaysburg *et al*.	International Symposium on Software Testing and Analysis	2002

E	Deriving operational software specifications from system goals	Letier and van Lamsweerde	ACM SIGSOFT symposium on Foundations of software engineering	2002
NC	Deriving test cases for composite operations in Object-Z specifications	Periyasamy *et al*.	Technology of OO Languages and Systems (TOOLS 26)	1999
D	Deriving test plans from architectural descriptions	Bertolino *et al.*	ICSE	2000
A	Deriving tests from UML 2.0 sequence diagrams with neg and assert	Lund and Stølen	AST	2006
D	Design and implementation of Triveni: a process-algebraic API for threads + events	Colby *et al*.	International Conference on Computer Languages	1998
D	Developing a TTCN-3 test harness for legacy software	Okika *et al*.	AST	2006
D	Distributed software testing with specification	Chang et al.	IEEE Computer Society's International Computer Software & Applications Conference	1990
B	Domain specific test case generation using higher ordered typed languages for specification	Sinha and Smidts	University of Maryland at College Park	2005
E	Early estimation of defect density using an in-process Haskell metrics model	Mark Sherriff	A-MOST	2005
E	Economic perspectives in test automation: balancing automated and manual testing with opportunity cost	Ramler and Wolfmaier	AST	2006
E	Engineering with logic: HOL specification and symbolic-evaluation testing for TCP implementations	Bishop *et al*.	Annual ACM Symposium on Principles of Programming Languages	2006
E	Enhanced testing of domain specific applications by automatic extraction of axioms from functional specifications	Sinha *et al*.	ISSRE	2003
B	Environment behavior models for scenario generation and testing automation	Auguston *et al*.	A-MOST	2005
B	Evaluating several path-based partial dynamic analysis methods for selecting black-box generated test cases	Chan and Yu	QSIC	2004
E	Experience With Teaching Black-Box Testing in a Computer Science/Software Engineering Curriculum	Chen and Poon	IEEE Transactions on Education	2004
E	Experimental Modal Analysis and Computational Model Updating of a Car Body in White	Schedlinski *et al*.	International Conference on Noise and Vibration Engineering	2004
NC	Extended model-based testing toward high code coverage rate	Takahashi and Kakuda	SOFTWARE QUALITYECSQ	2002
E	Extending Simulink Models With Natural Relations To Improve Automated Model-Based Testing	Boden *et al*.	SEW	2005
D	Extending test templates with inheritance	Murray *et al*.	ASWEC	1997
E	Fault classes and error detection capability of specification-based testing	Kuhn	ACM Transactions on Software Engineering and Methodology	1999
E	Fault model-driven test derivation from finite state models	Petrenko	Modeling and verification of parallel processes	2001
NC	Feature interaction detection using a synchronous approach and testing	du Bousquet *et al*.	Computer Networks	2000
E	Formal methods software engineering for the CARA system	Martin	International Journal on Software Tools for Technology Transfer	2004
NC	Formal Specification Based Software Testing: An Automated Approach	Gill and Bhatia	Proceedings of the International Conference on Software Engineering Research and Practise	2003
C	Formal test-case generation for UML statecharts	Gnesi et al.	IEEE ICECCS	2004
D	Formally testing fail-safety of electronic purse protocols	Jurjens and Wimmel	ASE	2001
B	From faults via test purposes to test cases: on the fault-based testing of concurrent systems	Aichernig e Delgado	9th International Conference on Fundamental Approaches to Software Engineering {FASE}	2006
D	From MC/DC to RC/DC: Formalization and analysis of control-flow testing criteria	Vilkomir and Bowen	Formal Aspects of Computing	2006
D	From Object-Z specifications to ClassBench test suites	Carrington et al.	Journal of Software Testing Verification and	2000

			Reliability	
E	From U2TP models to executable tests with TTCN-3: an approach to model driven testing	Zander *et al*.	Testing of communicating systems	2005
E	Generating a test oracle from program documentation: work in progress	Peters and Parnas	ACM SIGSOFT international symposium on Software testing and analysis	1994
B	Generating functional test cases in-the-large for time-critical systems from logic-based specifications	Morasca *et al*.	ISSTA	1996
E	Generating optimal distinguishing sequences with a model checker	Mallett *et al*.	A-MOST	2005
E	Generating oracles from your favorite temporal logic specifications	Dillon and Ramakrishna	ACM SIGSOFT Symposium on the Foundations of Software Engineering	1996
D	Generating regression tests via model checking	Lihua *et al*.	COMPSAC	2004
NC	Generating test case based on UML statecharts	Miao-Huai *et al*.	Mini Micro Systems	2005
B	Generating test cases for real-time systems from logic specifications	Mandrioli *et al*.	ACM Transactions on Computer Systems	1995
C	Generating Test Cases from an OO Model with an AI Planning System	Scheetz *et al*.	ISSRE	1999
B	Generating test cases from class vectors	Leung *et al*.	Journal of Systems and Software	2003
A	Generating test cases from UML activity diagram based on Gray-box method	Linzhang *et al*.	APSEC	2004
NC	Generating test data for specification-based tests via quasirandom sequences	Che *et al*.	Lecture Notes in Computer Science	2006
B	Generating test data from SOFL specifications	Offutt and Liu	Journal of Systems and Software	1999
A	Generating Test Data from State-Based Specifications	Offutt *et al*.	Journal of Software Testing, Verification and Reliability	2003
C	Generating Test Sequences from UML Sequence Diagrams and State Diagrams	Sokenou	Informatik Forschung und Entwicklung	2006
B	Generating test suites for software load testing	Avritzer and Weyuker	International symposium on Software testing and analysis	1994
A	Generating tests from UML specifications	Offutt and Abdurazik	UML	1999
E	Generating transition probabilities to support model-based software testing	Walton and Poore	Software - Practice and Experience	2000
B	Generating, selecting and prioritizing test cases from specifications with tool support	Yu *et al.*	QSIC	2003
NC	Generation of reliability test data with UML for real-time embedded software	Jun and Minyan	Journal of Beijing University of Aeronautics and Astronautics	2003
E	Generation of test sequences from formal specifications: GSM 11-11 standard case study	Bernard et al.	Software - Practice and Experience	2004
B	Identification of categories and choices in activity diagrams	Chen *et al*.	QSIC	2005
E	Implementation of model based intelligent next generation test generator using neural networks	Singer	SPIE The International Society for Optical Engineering	1996
D	Improving design dependability by exploiting an open model-based specification	Tomita and Sakamura	IEEE Transactions on Computers	1999
E	Improving functional testing using model-based diagnostics	Nolan and Carey	Proceedings of the Technical Program	1998
E	Improving functional/diagnostic testing using model-based reasoning	Carey and Dussault	IEEE AUTOTESTCON Proceedings	1998
C	Improving State-Based Coverage Criteria Using Data Flow Information	Briand *et al*.	Technical Report of Carleton University	2004
D	Improving test suites via operational abstraction	Harder *et al*.	ICSE	2003
B	Improving web application testing with user session data	Elbaum *et al*.	ICSE	2003
E	In this Issue	Briand and Basili	Empirical Software Engineering	2005
E	Increasing dependability by means of model-based acceptance test inside RTOS	Zhao *et al*.	Parallel Processing and Applied Mathematics	2005
B	In-parameter-order: a test generation strategy for pairwise testing	Lei and Tai	IEEE International High-Assurance Systems Engineering Symposium	1998
E	Integrating automated test generation into the WYSIWYT spreadsheet testing methodology	Fisher *et al*.	TOSEM	2006
E	Integrating formal specification and software verification and validation	Duke *et al*.	Teaching Formal Methods	2004
C	Integration of ‘components’ to test software components	Bertolino *et al*.	Electronic Notes in Theoretical Computer Science	2003

B	Integration of specification-based and CR-based approaches for GUI testing	Chen *et al*.	AINA	2005
C	Introducing a Reasonably Complete and Coherent Approach for Model-based Testing	Bertolino *et al*.	Electronic Notes in Theoretical Computer Science	2005
D	JUMBL: a tool for model-based statistical testing	Prowell	Annual Hawaii International Conference on System Sciences	2003
D	Korat: automated testing based on Java predicates	Boyapati et al.	ISSTA	2002
B	Lessons learned from automating tests for an operations support system	Fecko and Lott	Software—Practice & Experience	2002
D	Lutess: a specification-driven testing environment for synchronous software	du Bousquet *et al*.	ICSE	1999
E	Mars Polar Lander fault identification using model-based testing	Blackburn *et al*.	IEEE International Conference on Engineering of Complex Computer Systems	2002
E	Mars Polar Lander fault identification using model-based testing	Blackburn *et al*.	Annual NASA Goddard Software Engineering Workshop	2001
B	Mastering test generation from smart card software formal models	Bouquet *et al*.	Lecture Notes in Computer Science	2005
D	MaTeLo - statistical usage testing by annotated sequence diagrams, Markov chains and TTCN-3	Dulz and Zhen	International Conference on Quality Software	2003
E	Micro architecture coverage directed generation of test programs	Ur and Yadin	ACM/IEEE conference on Design automation	1999
NC	Model based development of embedded vehicle software at DaimlerChrysler	Conrad *et al*.	Informatik Forschung und Entwicklung	2005
D	Model based regression test reduction using dependence analysis	Korel *et al*.	ICSM	2002
B	Model based testing in evolutionary software development	Pretschner *et al*.	RSP	2001
NC	Model based testing in incremental system development	Pretschner *et al*.	Journal of Systems and Software	2004
B	Model Based Testing in Practice at Microsoft	Stobie	Electronic Notes in Theoretical Computer Science	2005
E	Model checking	Berg and Raffelt	Model Based Testing of Reactive Systems	2004
E	Model-Based Approaches for Validating Business Critical Systems	Augusto *et al*.	STEP	2003
C	Model-Based Built-In Tests	Gross *et al*.	Electronic Notes in Theoretical Computer Science	2005
B	Model-based formal specification directed testing of abstract data types	Jia	COMPSAC	1993
D	Model-based functional conformance testing of web services operating on persistent data	Sinha and Pardkar	TAV-WEB	2006
B	Model-based specification and testing applied to the Ground-Based Midcourse Defense (GMD) system: an industry report	Lakey	A-MOST	2005
NC	Model-based system testing of software product families	Reuys *et al*.	Lecture Notes in Computer Science	2005
B	Model-based test case generation for smart cards	Philipps *et al*.	Electronic Notes in Theoretical Computer Science	2003
E	Model-Based Test Driven Development of the Tefkat Model-Transformation Engine	Steel and Lawley	ISSRE'	2004
NC	Model-based test generation and implementation	Hartman and Proeyen	Informatie	2003
E	Model-based testing	Pretschner	ICSE	2005
A	Model-based testing and maintenance	Deng *et al*.	*ICMSE*	*2004*
E	Model-based Testing Considering Cost, Reliability and Software Quality	Htoon and Thein	APSITT	2005
A	Model-based testing for applications derived from software product lines	Olimpiew and Gomaa	A-MOST	2005
E	Model-based testing for enterprise software solutions	Jain	COMPSAC	2005
B	Model-based testing for real : the inhouse card case study	Pretschner *et al*.	International Journal on Software Tools for Technology Transfer (STTT)	2001
A	Model-Based Testing from UML Models	Bernard *et al*.	Informatik Forschung und Entwicklung	2006
B	Model-based testing in practice	Dalal *et al*.	ICSE	1999
B	Model-based testing of a highly programmable system	Dalal *et al*.	International Symposium on Software Reliability Engineering	1998
E	Model-based testing of concurrent programs with predicate sequencing constraints	Wu and Lin	QSIC	2005

A	Model-based testing of object-oriented systems	Rumpe	Formal Methods for Components and Objects	2003
NC	Model-based testing through a GUI	Kervinen *et al*.	FATES	2005
C	Model-based testing with UML applied to a roaming algorithm for Bluetooth devices	Zhen *et al*.	Journal of Zhejiang University Science	2004
E	Model-Based Testing: Challenges Ahead	Heimdahl	COMPSAC	2005
E	Model-based tests of truisms	Menzies *et al*.	ASE	2002
NC	Modeling and testing agent systems based on statecharts	Seo *et al*.	FORTE Workshops	2004
B	Modeling requirements for combinatorial software testing	Lott *et al*.	A-MOST	2005
E	Modellgestutzte Erprobungsmethodik in der Antriebsstrangentwicklung	Albers and Schyr	VDI Berichte	2005
E	Modelling the quality economics of defect-detection techniques	Wagner	WoSQ	2006
B	Models for synchronous software testing	Lakehal *et al*.	International Workshop on Model, Design and Validation	2004
E	Multiplexing of partially ordered events	Campbell *et al*.	Lecture Notes in Computer Science	2005
E	Mutation operators for Object-Z specification	Liu and Miao	ICECCS	2005
E	Mutation Testing Applied to Estelle Specifications	Souza *et al*.	Software Quality Control	1999
B	Mutation-based testing criteria for timeliness	Nilsson *et al*.	COMPSAC	2004
E	Mutually enhancing test generation and specification inference	Tao and Notkin	Formal Approaches to Software Testing	2004
E	Non-specification-based approaches to logic testing for software	Kobayashi *et al*.	Information and Software Technology	2002
E	On fault classes and error detection capability of specification-based testing	Tsuchiya and Kikuno	ACM Transactions on Software Engineering and Methodology	2002
B	On the Complexity of Generating Optimal Test Sequences	Boyd and Ural	IEEE Transactions on Software Engineering	1991
E	On the economics of requirements-based test case prioritization	Srikanth and Williams	EDSER	2005
NC	On the effectiveness of classification trees for test case construction	Chen and Poon	Information and Software Technology	1998
B	On the effectiveness of mutation analysis as a black box testing technique	Murnane and Reed	ASWEC	2001
E	On the identification of categories and choices for specification-based test case generation	Chen *et al*.	Information and Software Technology	2004
B	On the integration of design and test: a model-based approach for embedded systems	Pfaller *et al*.	AST	2006
E	On the relationships of faults for Boolean specification based testing	Lau and Yu	ASWEC	2001
E	On the State of the Art in Requirements-based Validation and Test of Software	Ryser *et al*.	Techinical Report at University of Zurich	1999
E	On the testing methods used by beginning software testers	Yu *et al*.	Information and Software Technology	2004
E	On the use of the classification-tree method by beginning software testers	Yu *et al*.	ACM symposium on Applied computing	2003
E	One evaluation of model-based testing and its automation	Pretschner *et al*.	ICSE	2005
E	On-line Fault Detection of Sensor Measurements	Koushanfar *et al*.	Proceedings of IEEE Sensors	2003
B	Online testing with model programs	Veanes *et al*.	ESEC/FSE	2005
B	Optimal strategies for testing nondeterministic systems	Lev Nachmanson	ISSTA	2004
D	Parameterized unit tests	Tillmann and Schulte	ESEC/FSE-13	2004
B	Plannable test selection criteria for FSMs extracted from operational specifications	Paradkar	ISSRE	2004
D	Play to test	Blass *et al*.	Formal Approaches to Software Testing	2005
D	Practical approach to specification and conformance testing of distributed network applications	Kuliamin *et al*.	Lecture Notes in Computer Science	2005
NC	Preamble computation in automated test case generation using constraint logic programming	Colin *et al.*	Software Testing, Verification & Reliability	2004
E	Prioritizing JUnit Test Cases: An Empirical Assessment and Cost-Benefits Analysis	Do *et al.*	Empirical Software Engineering	2006
B	Probe: a formal specification-based testing system	Amayreh and Zin	International conference on Information Systems	1999
E	Product family testing: a survey	Tevanlinna *et al*.	ACM SIGSOFT Software Engineering Notes	2004
B	Projected state machine coverage for software testing	Friedman *et al*.	ISSTA	2002
E	Property-based testing: a new approach to testing for assurance	Fink and Bishop	ACM SIGSOFT Software Engineering Notes	1997

B	ProTest: An Automatic Test Environment for B Specifications	Satpathy *et al*.	Electronic Notes in Theoretical Computer Science	2005
NC	proTEST: an automatic, scenario-driven, requirements-based software test process	Hirt	CONQUEST	2000
E	Random testing of interrupt-driven software	Regehr	EMSOFT	2005
NC	Refinement in statechart testing	Bogdanov and Holcombe	Software Testing Verification & Reliability	2004
D	Regression testing of classes based on TCOZ specification	Liang	ICECCS	2005
B	Requirement-based automated black-box test generation	Tahat *et al*.	COMPSAC	2001
B	Requirements traceability in automated test generation: application to smart card software validation	Bouquet *et al*.	A-MOST	2005
B	Requirements-Based Monitors for Real-Time Systems	Peters and Parnas	IEEE Transactions on Software Engineering	2002
E	Requirements-driven software test: a process-oriented approach	Ramachandran	ACM SIGSOFT Software Engineering Notes	1996
NC	Reusing class-based test cases for testing object-oriented framework interface classes	Dallal and Sorenson	Journal of Software Maintenance and Evolution: Research and Practice	2005
E	Reverse engineering of test cases for selective regression testing	Sneed	European Conference on Software Maintenance and Reengineering	2004
C	Revisiting Strategies for Ordering Class Integration Testing in the Presence of Dependency Cycles – An Investigation of Graph-based Class Integration Test Order Strategies	Briand *et al*.	Technical Report of Carleton University	2002
D	SALT - An Integrated Environment to Automate Generation of Function Tests for APIs	Paradkar	ISSRE	2000
NC	Scenario-based system test with software-product families	Reuys *et al*.	Informatik Forschung und Entwicklung	2005
E	Selecting small yet effective set of test data	Paradkar	IASTED International Multi-Conference on Applied Informatics	2003
E	Separating sequence overlap for automated test sequence generation	Hierons	ASE	2006
NC	Sequencing constraints-based regression testing of concurrent programs after specification changes	Kim *et al*.	Journal of KISS: Software and Applications	2000
NC	Siddhartha - automated test driver-oracle synthesis	Richardson	ACM SIGSOFT Software Engineering Notes	2000
E	Siddhartha: a method for developing domain-specific test driver generators	Reyes and Richardson	ASE	1999
D	Software Architecture Analysis Based on Statechart Semantics	Dias and Vieira	International Workshop on Software Specification and Design	2000
E	Software assurance by bounded exhaustive testing	Sullivan *et al*.	ISSTA	2004
E	Software Fault Tolerance: A Tutorial	Wilfredo	NASA Langley Technical Report Server	2000
E	Software in Metrology	RICHTER	PTB-MITTEILUNGEN	1991
E	Software model checking in practice: an industrial case study	Chandra *et al*.	ICSE	2002
NC	Software requirements and acceptance testing	Hsia *et al*.	Annals of Software Engineering	1997
E	Software requirements validation via task analysis	Zhu *et al*.	Journal of Systems and Software	2002
E	Software test selection patterns and elusive bugs	Howden	COMPSAC	2005
D	Software testing at the architectural level	Richardson and Wolf	ISAW-2 and Viewpoints	1996
E	Software unit test coverage and adequacy	Zhu *et al*.	ACM Computing Surveys	1997
B	Specification based test sequence generation with propositional logic	Wimmel *et al*.	Software Testing Verification and Reliability	2000
NC	Specification testing of synchronous software	Parissis	Technique et Science Informatiques	2002
D	Specification-based class testing with ClassBench	Murray *et al*.	Asia Pacific Software Engineering Conference	1998
A	Specification-based regression test selection with risk analysis	Chen *et al*.	*CASCON*	*2002*
B	Specification-Based Test Generation for Security-Critical Systems Using Mutations	Wimmel and Jürjens	International Conference on Formal Methods and Software Engineering	2002
D	Specification-based test oracles for reactive systems	Richardson *et al*.	ICSE	1992
NC	Specification-based testing for GUI-based applications	Chen and	Software Quality Control	2002

		Subramaniam		
B	Specification-based testing for real-time avionic systems	Biberstein and Fitzgerald	IEE Colloquium on Applicable Modelling, Verification and Analysis Techniques for Real-Time Systems	1999
B	Specification-based testing for real-time reactive systems	Alagar et al.	TOOLS	2000
E	Specification-based testing of concurrent programs	Carver	ACM SIGSOFT Software Engineering Notes	2000
NC	Specification-based testing of concurrent systems	Ulrich and Konig	Formal Description Techniques and Protocol Specification, Testing and Verification	1998
B	Specification-based testing of reactive software: a case study in technology transfer	Jagadeesan *et al*.	Journal of Systems and Software	1998
B	Specification-based testing of reactive software: tools and experiments-experience report	Jagadeesan *et al*.	ICSE	1997
B	Specification-based testing of synchronous software	Parissis and Ouabdesselam	Symposium on the Foundations of Software Engineering	1996
NC	Specification-based testing of user interfaces	Paiva *et al*.	Interactive Systems, Design, Specification, and Verification	2003
B	Specification-based testing using cause-effect graphs	Paradkar *et al*.	Annals of Software Engineering	1997
B	Specification-based testing with linear temporal logic	Tan *et al*.	IRI	2004
D	Specifying and Testing Software Components using ADL	Hayes and Sankar	Techinical Report at Sun Microsystems	1994
D	State-based incremental testing of aspect-oriented programs	Xu and Xu	AOSD	2006
D	State-based testing of integration aspects	Xu and Xu	WTAOP	2006
B	Statechart testing method for aircraft control systems	Bogdanov and Holcombe	Software Testing Verification & Reliability	2001
E	Static driver verifier, a formal verification tool for Windows device drivers	Levin	MEMOCODE	2004
E	Status report: requirements engineering	Hsia *et al*.	IEEE Software	1993
B	Strategies for automated specification-based testing of synchronous software	Parissis and Vassy	ASE	2001
E	Structural coverage analysis method	Gifford	AIAA/IEEE Digital Avionics Systems Conference	1996
D	Structural specification-based testing with ADL	Chang *et al*.	ISSTA	1996
D	Structural specification-based testing: automated support and experimental evaluation	Chang and Richardson	ESEC	1999
NC	Study on software test method based on finite state machine	Lan *et al*.	Journal of North China Electric Power University	2005
E	Supporting Controlled Experimentation with Testing Techniques: An Infrastructure and its Potential Impact	Do *et al*.	Empirical Software Engineering	2005
NC	Systematic Model-Based Testing of Embedded Automotive Software	Conrad *et al*.	Electronic Notes in Theoretical Computer Science	2005
B	Systematic model-based testing of embedded control software: the MB^3/T approach	Conrad *et al*.	ICSE	2004
NC	Systematic testing as base for securing quality	Schlatter	Information Management and Consulting	2006
B	Telecommunication software validation using a synchronous approach	du Bousquet *et al*.	ASSET	1998
B	Test Case Generation as an AI Planning Problem	Howe *et al*.	ASE	1997
C	Test cases generation from UML state diagrams	Kim *et al*.	IEE Software	1999
D	Test input generation for java containers using state matching	Visser *et al*.	ISSTA	2006
D	Test input generation with java PathFinder	Visser *et al*.	ISSTA	2004
E	Test prioritization for pairwise interaction coverage	Bryce and Colbourn	A-MOST	2005
A	Test ready UML statechart models	Murthy *et al*.	SCESM	2006
D	Test selection for object-oriented software based on formal specifications	Peraire *et al*.	International Conference on Programming Concepts and Methods	1998
A	Test selection from UML Statecharts	Liuying and Zhichang	TOOLS	1999

NC	Test specification based on trace description	Petrenko	Programming and Computer Software	1993
D	Test template framework: a specification-based testing case study	Stocks and Carrington	ISSTA	1993
D	Test templates: a specification-based testing framework	Stocks and Carrington	ICSE	1993
B	Test-based model generation for legacy systems	Hungar *et al*.	ITC	2003
D	TestEra: Specification-based testing of java programs using SAT	Khurshid and Marinov	ASE	2004
E	Testing against some eventuality properties of synchronous software: A case study	du Bousquet *et al*.	Electronic Notes in Theoretical Computer Science	2004
A	Testing agile requirements models	Botaschanjan *et al*.	Journal of Zhejiang University Science	2004
B	Testing of concurrent programs based on message sequence charts	Chung *et al*.	International Symposium on Software Engineering for Parallel and Distributed Systems	1999
E	Testing polymorphic interactions in UML sequence diagrams	Supavita and Suwannasart	ITCC	2005
B	Testing real-time embedded software using UPPAAL-TRON: an industrial case study	Larsen *et al*.	EMSOFT	2005
E	Testing software modelling tools using data mutation	Shan	AST	2006
B	Testing times: On Model-Driven Test Generation for Non-Deterministic Real-Time Systems	Brinksma, E.	ACSD	2004
B	Testing web applications by modeling with FSMs	Andrews *et al*.	Systems and Modeling	2005
E	Testing with model checker: insuring fault visibility	Okun *et al*.	WSEAS Transactions on Systems	2003
B	Test-suite reduction for model based tests: effects on test quality and implications for testing	Heimdahl and George	ASE	2004
C	TestUml: user-metrics driven web applications testing	Bellettini *et al*.	SAC	2005
E	The (Im)maturity level of software testing	Bertolino	ACM SIGSOFT Software Engineering Notes	2004
E	The advanced mobile application testing environment	Binder and Hanlon	A-MOST	2005
A	The AGEDIS tools for model based testing	Hartman and Nagin	ISSTA	2004
E	The ASTOOT approach to testing object-oriented programs	Doong and Frankl	TOSEM	1994
E	The effect of code coverage on fault detection under different testing profiles	Cai and Lyu	A-MOST	2005
E	The semantics of triveni: A process-algebraic API for threads + events	Colby *et al*.	Electronic Notes in Theoretical Computer Science	1998
E	The specification-based testing of a trusted kernel: MK++	Ford *et al*.	International Conference Conference on Formal Engineering Methods	1997
E	The state problem for test generation in Simulink	Zhan and Clark	GECCO	2006
E	The UniTesK Approach to Designing Test Suites	Kuliamin *et al*.	Programming and Computing Software ,	2003
E	The use of model-based test requirements throughout the product life cycle	Bukata *et al.*	Aerospace and Electronic Systems Magazine	2000
B	Thoroughness of specification-based testing of synchronous programs	Parissis and Vassy	ISSRE	2003
D	TinMan - A test derivation and management tool for specification-based class testing	Murray *et al*.	TOOLS	1999
E	Too darned big to test	Stobie	Queue	2005
E	Tools for model-based security engineering	Jürjens and Fox	ICSE	2006
B	Towards a more efficient way of generating test cases: class graphs	Leung and Wong	Asia-Pacific Conference on Quality Software	2000
C	Towards Automated Support for Deriving Test Data from UML Statecharts	Briand *et al*.	Technical Report of Carleton University	2004
B	Towards integration of use case modelling and usage-based testing	Regnell *et al*.	Journal of Systems and Software	2000
B	Towards model-based generation of self-priming and self-checking conformance tests for interactive systems	Paradkar	Information and Software Technology	2004
E	Towards model-driven testing of a Web application generator	Baresi et al.	Lecture Notes in Computer Science	2005
E	Traceability techniques: Using scenarios to support traceability	Naslavsky *et al*.	TEFSE	2005
C	Traffic-aware stress testing of distributed systems based on UML models	Garousi *et al*.	ICSE	2006

E	Translating test programs using a model-based process	Bearse and Lynch	AUTOTESTCON	1999
B	T-UPPAAL: online model-based testing of real-time systems	Mikucionis *et al*.	ASE	2004
B	T-VEC: a tool for developing critical systems	Blackburn and Busser	COMPASS	1996
B	T-VECTM product summary		Workshop on Industrial Strength Formal Specification Techniques	1998
NC	UML based statistical testing acceleration of distributed safety-critical software	Yan *et al*.	ISPA	2004
C	UML-based integration testing	Hartmann *et al*.	ISSTA	2000
C	UML-Based Integration Testing for Component-Based Software	Wu *et al*.	ICCBSS	2003
A	UML-based statistical test case generation	Riebisch *et al*.	NetObjectDays	2002
E	Uniform descriptions for model based testing	Krishnan	ASWEC	2004
E	Usage model-based automated testing of C++ templates	Sayre	A-MOST	2005
B	Use Case-based Testing of Product Lines	Bertolino and Gnesi	ESEC and FSE-11	2003
E	Use of sequencing constraints for specification-based testing of concurrent programs	Carver and Tai	IEEE Transactions on Software Engineering	1998
B	Use-case driven test for object-oriented system	Choi	Software Engineering and Applications	2001
E	Using a model checker to test safety properties	Ammann *et al.*	IEEE International Conference on Engineering of Complex Computer Systems	2001
B	Using artificial life techniques to generate test cases for combinatorial testing	Shiba *et al*.	International Computer Software and Applications Conference	2004
B	Using formal methods to derive test frames in category-partition testing	Ammann and Offutt	COMPASS	1994
B	Using formal specifications as test oracles for system-critical software	Hagar and Bieman	ACM SIGAda Ada Letters	1996
E	Using information about functions in selecting test cases	Clermont and Parnas	A-MOST	2005
B	Using model checking to generate tests from requirements specifications	Gargantini and Heitmeyer	ESEC/FSE-7	1999
E	Using model-based test program generator for simulation validation	Zhang *et al*.	Lecture Notes in Computer Science	2005
E	Using Simulation to Empirically Investigate Test Coverage Criteria Based on Statechart	Briand *et al.*	ICSE	2004
NC	Using state diagrams to generate unit tests for object-oriented systems	Ipate and Holcombe	Lecture Notes in Computer Science	2005
E	Using Test Oracles Generated from Program Documentation	Peters and Parnas	IEEE Transactions on Software Engineering	1998
E	Using the incremental approach to generate test sets: a case study	Yu *et al*.	QSIC	2003
C	Using UML Collaboration Diagrams for Static Checking and Test Generation	Abdurazik and Offutt	International Conference of UML	2000
C	Using UML for Automatic Test Generation	Crichton *et al*.	ASE	2001
B	Using Z specifications in category partition testing	Amla and Ammann	COMPASS	1992
E	Validation in model-driven engineering: testing model transformations	Fleurey *et al*.	International Workshop on Model, Design and Validation	2004
E	Validation test of distributed program based on event sequencing constraints	Qing *et al*.	Journal of Software	2000
A	Verification of requirements for safety-critical software	Carpenter	SIGAda	*1999*
B	White on black: a white-box-oriented approach for selecting black box-generated test cases	Chen *et al.*	Asia-Pacific Conference on Quality Software	2000

AP A.3.2 Segunda Execução – Agosto/2009

Categoria	Título	Autores	Fonte	Ano
E	A brief history of A-MOST Special Issue containing selected papers from A-MOST 2008	Frantzen et al.	Journal of Logic and Algebraic Programming	2009
A	A case study for generating test cases from use cases	Gutierrez et al.	ICRCIS	2008

E	A comparative evaluation of tests generated from different UML diagrams	Kansomkeat et al.	SNPD	2008
D	A formal approach for functional and structural test case generation in multi-agent systems	Kissoum e Sahnoun	AICCSA	2007
E	A formal methodology to test complex heterogeneous systems	Rodriguez e Nunez	Lecture Notes in Computer Science	2007
D	A generalized model-based test generation method	Bonifacio	SEFM	2008
B	A Global Algorithm for Model-Based Test Suite Generation	Hessel et al.	Electronic Notes in Theoretical Computer Science	2007
E	A logic for assessing sets of heterogeneous testing hypotheses	Rodriguez et al.	Lecture Notes in Computer Science	2006
NC	A measurement framework for evaluating model-based test generation tools	Sinha et al.	IBM Systems Journal	2006
E	A method and tools for large scale scenarios	Hall	ASE	2008
C	A method for model based test harness generation for component testing	Rocha e Martins	Journal of the Brazilian Computer Society	2008
E	A methodology for automated test generation guided by functional coverage constraints at specification level	Laurent et al.	ASE	2006
A	A model based testing technique to test web applications using StateCharts	Reza et al.	ITNG	2008
B	A model-based statistical usage testing of communication protocols	Popovic et al.	Workshop on Engineering of Computer Based Systems	2006
A	A model-driven validation & verification environment for embedded systems	Gargantini et al.	ISIES	2008
E	A note on an anomaly in black-box testing	Huima	Lecture Notes in Computer Science	2006
E	A quest for appropriate software fault models: Case studies on fault detection effectiveness of model-based test generation techniques	Paradkar	Information and Software Technology	2006
C	A recursive colored Petri Nets semantics for AUML as base of test case generation	Kissoum e Sahnoun	AICCSA	2008
B	A statistical approach to model-based robustness testing	Popovic e Kovacevic	Workshop on Engineering of Computer Based Systems	2007
A	A subset of precise UML for model-based testing	Bouquet et al.	AMOST	2007
NC	Advances in automated source-level debugging of verilog designs	Peischl et al.	Studies in Computational Intelligence	2008
B	Aiding modular design and verification of safety-critical time-triggered systems by use of executable formal specifications	Sakurai et al.	ISHASE	2008
E	An approach for experimentally evaluating effectiveness and efficiency of coverage criteria for software testing	Gupta et al.	Journal on Software Tools for Technology Transfer	2008
B	An approach for specification-based test case generation for Web services	Hanna e Munro	AICCSA	2007
D	An automated testing experiment for layered embedded C code	Chetali e Nguyen	Journal on Software Tools for Technology Transfer	2009
A	An automatic execution system for web functional test base on modelling user's behaviour	Jia et al.	ISISE	2008
A	An evaluation of a model-based testing method for information systems	Santos-Neto et al.	Symposium on Applied Computing	2008
E	An evaluation of model checkers for specification based test case generation	Fraser e Gargantini	ICST	2009
B	An event-flow model of GUI-based applications for testing	Memon	Software Testing Verification and Reliability	2007
B	An Extension of the Classification-Tree Method for Embedded Systems for the Description of Events	Conrad e Krupp	Electronic Notes in Theoretical Computer Science	2006
E	Analysing the effectiveness of rule-coverage as a reduction criterion for test suites of grammar-based software	Hennessy e Power	Empirical Software Engineering	2008
D	Application of system models in regression test suite prioritization	Korel et al.	ICSM	2008
NC	Applying river formation dynamics to solve NP-complete problems	Rabanal et al.	Studies in Computational Intelligence	2009
E	Applying the knowledge stored in systems models to derive validation tools and environments	Yague e Garbajosa	ICCI	2007
NC	Approach for optimizing test suite based on testing requirement reduction	Zhang et al.	Ruan Jian Xue Bao/Journal of Software	2007
C	Architecting fault tolerance with exception handling: Verification and validation	Brito et al.	Journal of Computer Science and Technology	2009

C	Aspects-classes integration testing strategy: An incremental approach	Massicotte et al.	Lecture Notes in Computer Science	2006
E	Augmenting automatically generated unit-test suites with regression oracle checking	Xie	Lecture Notes in Computer Science	2006
B	Automated boundary test generation from JML specifications	Bouquet et al.	Lecture Notes in Computer Science	2006
B	Automated formal verification and testing of C programs for embedded systems	Kandl et al.	ISORC	2007
B	Automated functional conformance test generation for semantic web services	Paradkar et al.	ICWS	2007
E	Automated generation of positive and negative tests for parsers	Zelenov e Zelenova	Lecture Notes in Computer Science	2006
A	Automated large-scale simulation test-data generation for object-oriented software systems	Zheng et al.	ISDPE	2007
B	Automatic generation of model based tests for a class of security properties	Masson et al.	AMOST	2007
E	Automatic generation of test models for model transformations	Wang et al.	ASWEC	2008
C	Automatic Model-Based Generation of Parameterized Test Cases Using Data Abstraction	Calame et al.	Electronic Notes in Theoretical Computer Science	2007
E	Automatic test case generation through a collaborative web application	Arantes et al.	IASTED	2008
C	Automatic test generation on a (U)SIM smart card	Bigot et al.	Lecture Notes in Computer Science	2006
D	Automatic testing from formal specifications	Satpathy et al.	Lecture Notes in Computer Science	2007
B	Automatic timed test case generation for web services composition	Lallali et al.	ECOWS	2008
D	Automatic validation of java page flows using model-based coverage criteria	Alava et al.	ICSAC	2006
E	Automation of broad sanity test generation	Zybin et al.	Programming and Computer Software	2008
E	Boundary value analysis using divide-and-rule approach	Vij e Fend	ITNG	2008
NC	Bridge the gap between software test process and business value: A case study	Li et al.	Lecture Notes in Computer Science	2009
E	Checking sequences for distributed test architectures	Hierons et al.	Distributed Computing	2008
E	Choosing a test modeling language: A survey	Hartman et al.	Lecture Notes in Computer Science	2007
NC	Combining different change prediction techniques	Cabrero et al.	ICEIS	2008
NC	Combining scenario- and model-based testing to ensure POSIX compliance	Dadeau et al.	Lecture Notes in Computer Science	2008
B	Combining test case generation for component and integration testing	Benz	AMOST	2007
E	Comparison of five black-box testing methods for object-oriented software	Kwang e Eun	SERA	2006
E	Computing refactorings of state machines	Pretschner et al.	Software and Systems Modeling	2007
C	Conformance testing based on UML state machines: Automated test case generation, execution and evaluation	Seifert	Lecture Notes in Computer Science	2008
E	Contract driven development = test driven development: Writing test cases	Leitner et al.	ESEC/FSE	2007
E	Coverage metrics to measure adequacy of black-box test suites	Rajan	ASE	2006
E	Data mining static code attributes to learn defect predictors	Menzies et al.	IEEE Transactions on Software Engineering	2007
E	Derivation of a suitable finite test suite for customized probabilistic systems	Llana-Diaz et al.	Lecture Notes in Computer Science	2006
B	Derivation of tests from timed specifications according to different coverage criteria	Merayo e Nunez	ICONS	2008
E	Designing and comparing automated test oracles for GUI-based software applications	Xe e Memon	ACM Transactions on Software Engineering and Methodology	2007
D	Designing fault injection experiments using state-based model to test a space software	Ambrosio et al.	Lecture Notes in Computer Science	2007
B	Development of a framework for automated systematic testing of safety-critical embedded systems	Kandl et al.	WISES	2006
B	Efficient software test case generation using genetic algorithm based graph theory	Rajappa et al.	ICETET	2008
E	Efficient solving of structural constraints	Elkarablieh et al.	ISSTA	2008
E	Efficiently generating structurally complex inputs with thousands of objects	Elkarablieh et al.	Lecture Notes in Computer Science	2007
B	Employing user profiles to test a new version of a GUI component in its context of use	Memon	Software Quality Journal	2006
E	Enforcing different contracts in hierarchical component-based systems	Collet et al.	Lecture Notes in Computer Science	2006

D	Extended finite state machine based test derivation driven by user defined faults	El-Fakih et al.	ICST	2008
B	Extending EFSMs to specify and test timed systems with action durations and time-outs	Merayo et al.	IEEE Transactions on Computers	2008
B	Extending EFSMs to specify and test timed systems with action durations and timeouts	Merayo et al.	Lecture Notes in Computer Science	2006
B	Extending Stream X-machines to specify and test systems with timeouts	Merayo et al.	SEFM	2008
E	Flexibility in modeling languages and tools: A call to arms	Van Wyk e Heimdahl	Journal on Software Tools for Technology Transfer	2009
E	Formal methods in industrial software standards enforcement	Grinevich et al.	Lecture Notes in Computer Science	2007
A	Formal test generation from UML models	Buchs et al.	Lecture Notes in Computer Science	2006
E	Formally comparing user and implementer model-based testing methods	Andres et al.	ICSTW'08	2008
E	Formally transforming user-model testing problems into implementer-model testing problems and viceversa	Andres et al.	Journal of Logic and Algebraic Programming	2009
NC	From MC/DC to RC/DC: Formalization and analysis of control-flow testing criteria	Vilkomir et al.	Lecture Notes in Computer Science	2008
B	Fully automatic testing with functions as specifications	Koopman e Plasmeijer	Lecture Notes in Computer Science	2006
NC	Generating tests from B specifications and test purposes	Julliand et al.	Lecture Notes in Computer Science	2008
D	Generating tests from EFSM models using guided model checking and iterated search refinement	Ernits et al.	Lecture Notes in Computer Science	2006
E	HOTL: Hypotheses and observations testing logic	Rodriguez et al.	Journal of Logic and Algebraic Programming	2008
B	HOTTest: A model-based test design technique for enhanced testing of domain-specific applications	Sinha et al.	ACM Transactions on Software Engineering and Methodology	2006
E	Improving evidence about software technologies: A look at model-based testing	Dias-Neto et al.	IEEE Software Magazine	2008
E	Improving fault detection capability by selectively retaining test cases during test suite reduction	Jeffrey e Gupta	IEEE Transactions on Software Engineering	2007
E	Incorporating varying requirement priorities and costs in test case prioritization for new and regression testing	Ramasamy et al.	ICCCN	2008
E	Industrial evaluation of a log file analysis methodology	Yantzi e Andrews	WODA	2007
E	Input parameter modeling for combination strategies	Grindal et al.	IASTED	2007
A	ISDGen: An automated simulation data generation tool for object-oriented information systems	Zheng et al.	ICSC	2007
E	Java-based automated test-framework [Java-basiertes automatisiertes Test-Framework]	Kalenborn et al.	Wirtschaftsinformatik	2006
A	Less is more: A minimalistic approach to UML model-based conformance test generation	Kaplan et al.	ICST	2008
NC	Maintenance cost of a software design: A value-based approach	Cabrero et al.	ICEIS	2007
B	Making model-based testing more agile: A use case driven approach	Katara e Kervinen	Lecture Notes in Computer Science	2007
E	Managing conflicts when using combination strategies to test software	Grindal et al.	ASWEC	2007
B	Message confidentiality testing of security protocols - Passive monitoring and active checking	Shu e Lee	Lecture Notes in Computer Science	2006
E	Metrics for model driven requirements development	Berenbach e Borotto	ICSE	2006
A	Model based testing of system requirements using UML use case models	Hasling et al.	ICST	2008
NC	Model based testing with labelled transition systems	Tretmans	Lecture Notes in Computer Science	2008
A	Model-based Security Testing Using UMLsec. A Case Study	Jurjens	Electronic Notes in Theoretical Computer Science	2008
B	Model-based security vulnerability testing	Pari-Salas et al.	ASWEC	2007
D	Model-based test prioritization heuristic methods and their evaluation	Korel et al.	AMOST	2007
B	Model-based test selection for infinite-state reactive systems	Jeannet et al.	Lecture Notes in Computer Science	2007
NC	Model-based test suite reduction with concept lattice	Ng e Fung	ASEA	2008
B	Model-based testing of thin-client web applications and navigation input	Koopman et al.	Lecture Notes in Computer Science	2007

B	Model-based tests for access control policies	Pretschner et al.	ICST	2008
E	Model-driven test-case construction	Baerisch	ESEC-FSE'07	2007
B	Modeling Web browser interactions and generating tests	Song et al.	CIS	2008
E	Models and software model checking of a Distributed File Replication system	Bjorner	Lecture Notes in Computer Science	2007
D	Module documentation based testing using grey-box approach	Baharom et al.	ITSim	2008
E	Multiple-view modelling and meta-modelling of software product lines	Gomaa et al.	IET Software	2008
E	Multi-view methodology for the design of embedded mechatronic control systems	Groothuis et a.	ICACSD	2007
E	Mutation analysis testing for model transformations	Mottu et al.	Lecture Notes in Computer Science	2006
E	Mutation testing in UTP	Aichernig e Jifeng	Formal Aspects of Computing	2009
E	Novel Software Automated Testing System Based on J2EE	Pei et al.	Tsinghua Science and Technology	2007
E	On model typing	Steel	Software and Systems Modeling	2007
C	On testing UML statecharts	Massink et al.	Journal of Logic and Algebraic Programming	2006
D	On the correctness of upper layers of automotive systems	Botaschanjan et al.	Formal Aspects of Computing	2008
B	On the effect of test-suite reduction on automatically generated model-based tests	Heimdahl et al.	ASE	2007
E	On the order of test goals in specification-based testing	Fraser et al.	Journal of Logic and Algebraic Programming	2009
A	Ontology-based test modeling and partition testing of web services	Bai et al.	ICWS	2008
D	Parallel test generation and execution with Korat	Misailovic et al.	ESEC/FSE	2007
D	PKorat: Parallel generation of structurally complex test inputs	Siddiqui e Khurshid	ICST	2009
C	Polymorphism sequence diagrams Test Data Automatic Generation based on OCL	Zhou et al.	ICYCS	2008
E	Pseudo-exhaustive testing for software	Kuhn e Okun	SEW-30	2006
E	Qualifying input test data for model transformations	Fleurey et al.	Software and Systems Modeling	2009
E	Qualitative modeling for requirements engineering	Menzies e Richardson	SEW-30	2006
E	Random testing of formal software models and induced coverage	Owen et al.	RT	2006
NC	Random Vs. scenario-based Vs. fault-based testing an industrial evaluation of formal black-box testing methods	Weiglhofer e Wotawa	ENASE	2008
B	Randomized directed testing (REDIRECT) for simulink/Stateflow models	Satpathy et al.	EMSOFT	2008
E	Reducing the costs of bounded-exhaustive testing	Jagannath et al.	Lecture Notes in Computer Science	2009
D	Redundancy based test-suite reduction	Fraser e Wotawa	Lecture Notes in Computer Science	2007
E	Refinement and Test Case Generation in UTP	Aichernig e He	Electronic Notes in Theoretical Computer Science	2007
D	Requirement model-based mutation testing for web service	Wang e Huang	NWeSP	2008
E	Requirement modeling for the C-5 modernization program	Allen et al.	CrossTalk	2009
E	Research of component-based hybrid design pattern for real-time microkernel	Ma et al.	ICAT	2006
A	Rigorous vertical software system testing in IDE	Kwang e Eun	SERA	2007
E	Safety and software intensive systems: Challenges old and new	Heimdahl	FoSE	2007
D	Security policy testing using vulnerability exploit chaining	Darmaillacq	ICSTW	2008
E	Software testing research: Achievements, challenges, dreams	Bertolino	FoSE	2007
D	Specification-based compaction of directed tests for functional validation of pipelined processors	Koo e Mishra	CODES	2008
E	Specification-based test generation and optimization using model checking	Zeng et al.	TASE	2007
B	Specification-based testing for software product lines	Kahsai et al.	ICSEFM	2008
C	Specification-based testing method using testing flow graphs	Voigt et al.	ICSEA	2007
E	Superfit combinational elusive bug detection	Barzin et al.	ICSAC	2008
NC	Survey on coverage directed generation technology	Shen et al.	Journal of Computer-Aided Design and	2009

			Computer Graphics	
B	Symbolic Model-based Test Selection	Jeron	Electronic Notes in Theoretical Computer Science	2009
NC	Synthesis of monitors for real-time analysis of reactive systems	Auguston et al.	Lecture Notes in Computer Science	2008
B	Synthesis of scenario based test cases from B models	Satpathy et al.	Lecture Notes in Computer Science	2006
NC	Systematic test data generation for embedded software	Zander-Nowicka et al.	SERP	2008
B	Test case generation from formal models through abstraction refinement and model checking	Satpathy, M., Ramesh	AMOST	2007
NC	Test case prioritization based on test suite design information	Qu et al.	Jisuanji Xuebao/Chinese Journal of Computers	2008
E	Test conditions for fault classes in Boolean specifications	Kapoor e Bowen	ACM Transactions on Software Engineering and Methodology	2007
D	Test generation and execution for security rules in temporal logic	Darmaillacq et al.	ICSTW	2008
B	Test generation from security policies specified in Or-BAC	Li et al.	ICSAC	2007
B	Test suite reduction based on dependence analysis	Jourdan et al.	Lecture Notes in Computer Science	2006
E	Test-case prioritization with model-checkers	Fraser e Wotawa	IASTED	2007
E	Test-driven assessment of access control in legacy applications	Le-Traon et al.	ICST	2008
E	TestFilter: A statement-coverage based test case reduction technique	Khan et al.	INMIC	2006
E	Testing data types implementations from algebraic specifications	Gaudel et al.	Lecture Notes in Computer Science	2008
B	Testing finite state machines presenting stochastic time and timeouts	Merayo et al.	Lecture Notes in Computer Science	2007
B	Testing from a stochastic timed system with a fault model	Hierons et al.	Journal of Logic and Algebraic Programming	2009
NC	Testing from X-machine specifications	Bogdanov	Lecture Notes in Computer Science	2008
B	Testing security properties of protocol implementations - A machine learning based approach	Shu e Lee	ICSCS	2007
NC	The criteria of functional delay test quality assessment	Bareisa et al.	DSD	2007
E	The effect of program and model structure on MC/DC test adequacy coverage	Rajan et al.	ICSE	2008
E	The feasibility of automated feedback-directed specification-based test generation: A case study of a high-assurance operating system	Wber et al.	ISSRE	2008
C	Towards a tool supporting integration testing of aspect-oriented programs	Massicotte et al.	Journal of Object Technology	2007
D	Towards modularized verification of distributed time-triggered systems	Botaschanjan et al.	Lecture Notes in Computer Science	2006
A	Towards traceability of model-based testing artifacts	Naslavsky	AMOST	2007
E	Transforming and selecting functional test cases for security policy testing	Mouelhi et al.	ICST	2009
B	Using communication coverage criteria and partial model generation to assist software integration testing	Robinson-Mallett et al.	Software Quality Journal	2008
E	Using formal specifications to support testing	Hierons et al.	ACM Computing Surveys	2009
B	Using JML runtime assertion checking to automate metamorphic testing in applications without test oracles	Murphy et al.	ICST	2009
D	Using model checking to generate fault detecting tests	Gargantini	Lecture Notes in Computer Science	2007
E	Using model-checkers to generate and analyze property relevant test-cases	Fraser e Wotawa	Software Quality Journal	2008
NC	Using process simulation to assess the test design effort reduction of a model-based testing approach	Aranha e Borba	Lecture Notes in Computer Science	2008
E	Validating the Microsoft Hypervisor	Cohen	Lecture Notes in Computer Science	2006
E	Virus coevolution partheno-genetic algorithms for optimal sensor placement	Kang et al.	Advanced Engineering Informatics	2008
NC	Visualization of use cases through automatically generated activity diagrams	Gutierrez et al.	Lecture Notes in Computer Science	2008
A	Web application model recovery for user input validation testing	Li et al.	ICSEA	2007
E	When Model-based Testing Fails	Aichernig e George	Electronic Notes in Theoretical Computer Science	2006
NC	WSDL-based automated test data generation for web service	Chunyan et al.	ICSSE	2008

APÊNDICE B – PLANO DO ESTUDO EXPERIMENTAL PARA AVALIAÇÃO DE *PORANTIM*

Este apêndice descreve os instrumentos usados no estudo experimental realizado ao longo deste trabalho que avaliou a abordagem Porantim, de apoio à seleção de TTBMs.

AP B.1 Instrumentos

AP B.1.1 Formulário E0 – Caracterização de Participantes

Formulário E0 – Caracterização do Participante

Nome: _____

Grupo: _____ **Data:** _____

Objetivo: conhecer o histórico, conhecimento e experiência do participante
Acrônimo: TTBM = Técnica de Teste Baseado em Modelos

Questões Genéricas

1. Qual heurística/ técnica você usa quando está selecionando casos de teste para avaliar um programa?
2. Qual heurística você usa quando está selecionando TTBM(s) para desenvolver um conjunto de casos de teste para um programa?
3. Qual informação você acredita ser relevante quando está decidindo a(s) TTBM(s) para desenvolver o conjunto de casos de teste para um programa? (sobre o projeto, o problema a ser resolvido, etc.). Tente ser o mais específico possível quando estiver referenciando as informações que você gostaria de saber quando estiver fazendo a seleção de TTBM.
4. Quais os tipos de problemas você acredita que irá encontrar quando estiver selecionando TTBM(s) para usar em um projeto?
5. Quanto tempo você acredita que levaria o processo de seleção de TTBM(s) para um projeto? Quais variáveis você imagina que irá afetar esse tempo?
6. O que você acredita que irá aprender com este exercício?

    0. Conhecimento que nunca será utilizado
    1. Conhecimento que será utilizado raramente
    2. Conhecimento que será utilizado muitas vezes
    3. Conhecimento essencial

    Justifique sua resposta:

    _____

    _____

AP B.1.2 Formulário E1 – Execução do Exercício com o Esquema

Formulário E1 – Execução do Exercício/Esquema

Nome: _____

Grupo: _____ **Data:** _____

Objetivo: Medir os tempos e conclusões

Acrônimo: TTBM = Técnica de Teste Baseado em Modelos

Material

Verifique que você recebeu todos os instrumentos que você precisa para realizar este exercício. Você deve possuir:

- Um documento de caracterização do projeto, com o nome: _____

- Um documento de caracterização de 13 TTBMs

- Os Formulários E1, E3, E4, E5, E6, E7 e E8

Exercício

Os passos a serem seguidos são:

1. Escreva abaixo a hora em que você iniciou o exercício: ___: ___ (hora:minuto)
2. Para realizar este exercício relembre que você deve:

a. (opcional, desconte este tempo). Olhar as técnicas de teste disponíveis.

b. Ler a documentação do projeto.

c. Examinar uma a uma as técnicas de teste baseado em modelos disponíveis.

d. Decidir quais são as técnicas de teste baseado em modelos que você usaria para o projeto.

3. Hora em que você começou a examinar a documentação do projeto: ___:___ (hora:minuto)
4. Hora em que você terminou de examinar a documentação do projeto: ___:___ (hora:minuto)
5. Tempo gasto para realizar a seleção (sem considerar possíveis interrupções): _____ minutos
6. Hora em que você iniciou a seleção das TTBMs: ___:___ (hora:minuto)
7. Hora em que você terminou a seleção das TTBMs: ___:___ (hora:minuto)
8. Tempo gasto para realizar a seleção (não considere possíveis interrupções): _____ minutos
9. Escreva abaixo a hora em que você terminou o exercício: ___:___ (hora:minuto)
10. Tempo que você levou para fazer o exercício (não considere possíveis interrupções): _____ minutos

Conclusões

O objetivo destas conclusões é estudar a evolução das opiniões dos participantes sobre o problema de seleção de TTBMs enquanto estava fazendo este exercício.

1. O que você aprendeu com este exercício?

a. Sobre o tipo de software a ser testado.

b. Sobre as características do ambiente do projeto.

c. Outros.

2. Após conhecer esta estratégia, a sua percepção sobre o problema de seleção de TTBMs mudou a respeito do tipo de: informações necessárias para a seleção, dificuldade e problemas encontrados?

a. Sobre o tipo de software a ser testado.

b. Sobre as características do ambiente do projeto.

c. Outros.

3. Se você tivesse que fazer este exercício novamente, usando novamente esta estratégia, você faria coisas diferentes? Quais? Por quê?

a. Sobre o tipo de software a ser testado.

b. Sobre as características do ambiente do projeto.

c. Outros.

AP B.1.3 Formulário E2 – Execução do Exercício com *Porantim*

Formulário E2 – Execução do Exercício/*Porantim*

Nome: _____

Grupo: _____ **Data:** _____

Objetivo: Medir os tempos e conclusões

Acrônimo: TTBM = Técnica de Teste Baseado em Modelos

Material

Verifique que você recebeu todos os instrumentos que precisa para realizar este exercício:

- Um documento de caracterização do projeto, com o nome: _____

- Um documento de caracterização de 13 TTBMs

- Os Formulários E1, E2, E4, E5, E6, E7, E8 e E9

Exercício

Os passos a serem seguidos são:

1. Escreva abaixo a hora em que você iniciou o exercício: ___: ___ (hora:minuto)
2. Para realizar este exercício relembre que você deve:

d. (opcional, desconte este tempo). Olhar as técnicas de teste disponíveis.

e. Ler a documentação do projeto.

f. Examinar uma a uma as técnicas de teste baseado em modelos disponíveis.

g. Decidir quais são as técnicas de teste baseado em modelos que você usaria para o projeto.

3. Hora em que você começou a examinar a documentação do projeto: ___:___ (hora:minuto)
4. Hora em que você terminou de examinar a documentação do projeto: ___:___ (hora:minuto)

5. Tempo gasto examinando a documentação do projeto (sem considerar interrupções): _____ minutos
6. Hora em que você iniciou a seleção das TTBMs: ___:___ (hora:minuto)
7. Hora em que você terminou a seleção das TTBMs: ___:___ (hora:minuto)
8. Quanto tempo você levou para realizar a seleção? (não considere possíveis interrupções) _____ minutos
9. Escreva abaixo a hora em que você terminou o exercício: ___:___ (hora:minuto)
10. Quanto tempo você levou para fazer o exercício? (não considere possíveis interrupções) _____ minutos

Conclusões

O objetivo destas conclusões é estudar a evolução das opiniões dos participantes sobre o problema de seleção de TTBMs enquanto estava fazendo este exercício.

1. O que você aprendeu com este exercício?

a. Sobre o tipo de software a ser testado.

b. Sobre as características do ambiente do projeto.

c. Outros.

2. Após conhecer esta estratégia, a sua percepção sobre o problema de seleção de TTBMs mudou a respeito do tipo de: informações necessárias para a seleção, dificuldade e problemas encontrados?

a. Sobre o tipo de software a ser testado.

b. Sobre as características do ambiente do projeto.

c. Outros.

3. Se você tivesse que fazer este exercício novamente, usando novamente esta estratégia, você faria coisas diferentes? Quais? Por quê?

a. Sobre o tipo de software a ser testado.

b. Sobre as características do ambiente do projeto.

c. Outros.

AP B.1.4 Formulário E3 – Informações usadas durante a Seleção

Formulário E3 – Informações Usadas durante a Seleção

Objetivo: Refletir as informações usadas sobre **todas** as TTBMs durante o processo de seleção

Nome: _____ **Página** _____ **de** _____

Nome do Documento: _____

Nº da TTBM	Informações/Atributos (você pode indicar o ID do(s) atributo(s))	Comentários

AP B.1.5 Formulário E4 – Informações não encontradas durante a Seleção

Formulário E4 – Informações não encontradas durante a Seleção

Objetivo: Refletir as informações que você sentiu falta sobre uma TTBM durante o processo de seleção

Nome: _____ **Página** _____ **de** _____

Nome do Documento: _____

Nº da TTBM	Informação	Comentários

AP B.1.6 Formulário E5 – Problemas/Dúvidas Encontrados durante a Seleção

Formulário E5 – Problemas/Dúvidas Encontrados durante a Seleção

Objetivo: Enumerar os problemas/dúvidas encontrados durante o processo de seleção

Nome: _____ **Página** _____ **de** _____

Nome do Documento: _____

Nº do Problema/ Dúvida	Problema/ Dúvida	Nº da TTBM	Comentários

AP B.1.7 Formulário E6 – Estudo das TTBMs

Formulário E6 – Estudo das Técnicas de TBM

Objetivo: Refletir o tempo gasto com o estudo (leitura) de cada TTBM

Nome: _____ **Página** _____ **de** _____

Nome do Documento: _____

Nº da TTBM	Hora de Início	Hora de Término	Tempo Total

AP B.1.8 Formulário E7 – TTBM s Selecionadas

Formulário E7 – Técnicas de TBM Selecionadas

Objetivo: Enumerar as TTBMs que você selecionou para o projeto

Nome: _____ **Página** _____ **de** _____

Nome do Documento: _____

Nº da Técnica	Técnica de TBM	Comentários

AP B.1.9 Formulário E8 – Ordenação de TTBMs

Formulário E8 – Ordenação de Técnicas de TBM

Objetivo: Ordenar as TTBMs de acordo com a relevância que você julga para cada uma para o projeto

Nome: _____ **Página** _____ **de** _____

Nome do Documento: _____

Ordem	Nº da TTBM	Comentários
1º		
2º		
3º		
4º		
5º		
6º		
7º		
8º		
9º		
10º		
11º		
12º		
13º		

AP B.1.10 Formulário E9 – Avaliação de *Porantim*

Formulário E9 – Avaliação de *Porantim*

Nome: _____

Grupo: _____ **Data:** _____

Objetivo: Verificar o potencial que os participantes observam sobre *Porantim*

Sobre *Porantim*

O objetivo dessa seção é conhecer a opinião dos participantes sobre a estratégia *Porantim* e estudar a evolução da opinião dos participantes sobre o problema de seleção de TTBMs durante o exercício.

1. Você acha que *Porantim* facilita a seleção de TTBMs? Por quê?
2. Você acha que *Porantim* é simples de ser entendida?
3. Você acha que o nome dos atributos usados por *Porantim* possam ser melhorados?
4. Qual o processo que você seguiu para usar *Porantim*? Caso nenhum tenha sido recomendado, por favor, explique-o.
5. Se você tivesse que usar *Porantim*, você acha que seria simples usá-la? Você acredita que *Porantim* possa ser melhorada para tornar seu uso mais simples? Como?
6. Como você observou, a utilização de *Porantim* implica no investimento de um tempo adicional para que todas as informações usadas possam estar disponíveis durante a seleção, ou seja, a caracterização mais detalhada das TTBMs. Levando-se isso em consideração, se você tivesse a oportunidade de usar *Porantim* em seu trabalho, você faria isso?
7. O que você aprendeu sobre a estratégia *Porantim*?

AP B.2 Repositório de TTBMs usado no Estudo Experimental

A seguir são apresentadas as caracterizações das TTBMs utilizadas no experimento. As características usadas são utilizadas pelas duas abordagens de seleção utilizadas no experimento (*Porantim* e Esquema de Caracterização). Característica Tipos de Falhas seguirá a taxonomia apresentada em SANTHANAM e CHILLAREGE (1998) composta por 13 diferentes tipos de falhas:

- Código faltando;
- Erro de Navegação;
- Erro de Interface;
- Erro de arquivo de dados;
- Erro do ambiente;
- Erro do banco de dados;
- Erro de Shell Script;
- Erro de exceção;
- Lógica Incorreta no código;
- Tipo de dados incorreto

AP B.2.1 Técnica 1: Abdurazik e Offut (2000)

ABDURAZIK, A. OFFUTT, J. (2000), “Using UML Collaboration Diagrams for Static Checking and Test Generation”, Third International Conference The Unified Modeling Language (UML’2000), York, UK, October, 1939, 383-395.

Atributo de Caracterização	**Valores**	
	Porantim	**Esquema**
Plataforma de execução do software	Independente de plataforma de execução	
Paradigma de desenvolvimento de software	Orientado a objeto	
Linguagem de programação	Independente de linguagem de programação	
Nível(is) de Teste	Teste de Sistema	
Características de qualidade do software que podem ser avaliadas pela abordagem	Funcionalidade Eficiência (Tempo de Resposta)	
Ferramenta de Apoio	Plug-in para o Rational Rose	
Custo associado à ferramenta	Custo da aquisição da ferramenta Rational Rose	
Plataforma em que a ferramenta opera	Extrai modelos da ferramenta Rational Rose, mas pode usar modelos exportados por outras ferramentas	
Nível de Automação	Todos os passos são automatizados	
Complexidade dos passos não-automatizados	Simples, apenas a modelagem inicial é requerida	
Resultados gerados	Dados de teste a serem usados em casos de teste	
Tipos de Falhas que permite revelar	Erro de Navegação Erro de Banco de Dados Lógica Incorreta no código Tipo de Dados Incorreto	
Habilidade requerida para ser operada	Conhecimento sobre modelagem de sistemas orientados a objetos	
Critério de Geração dos Testes	Busca em grafo	
Modelo comportamental/estrutural	Diagrama de Colaboração	-
Tecnologia usada para modelagem	UML	-
Tipo de Técnica de Teste	Estrutural	-
Entradas Requeridas	Diagrama de Colaboração	-
Existência de um verificador de modelo	Sim	-
Existência de um mecanismo de rastreabilidade	Não	-
Compreensibilidade	-	Utiliza um modelo que segue o meta-modelo da UML
Dependências	-	Não depende de outra abordagem

AP B.2.2 Técnica 2: Briand e Labiche (2002)

BRIAND, L. C., LABICHE, Y. (2001), “A UML-Based Approach to System Testing”, Proceedings of the 4th international Conference on the Unified Modeling Language, Modeling Languages, Concepts, and Tools, vol. 2185, London, Outubro, pp. 194-208.

Atributo de Caracterização	**Valores**	
	Porantim	**Esquema**
Plataforma de execução do software	Independente de plataforma de execução	
Paradigma de desenvolvimento de software	Orientado a objetos	
Linguagem de programação	Independente de linguagem de programação	
Nível(is) de Teste	Teste de Sistema	
Características de qualidade do software que podem ser avaliadas pela abordagem	Funcionalidade	
Ferramenta de Apoio	TOTEM	
Custo associado à ferramenta	Gratuita	
Plataforma em que a ferramenta opera	Multi-plataforma	
Nível de Automação	Todos os passos são automatizados	
Complexidade dos passos não-automatizados	Simples, apenas a modelagem inicial é requerida	
Resultados gerados	Casos de Teste	
Tipos de Falhas que permite revelar	Código faltando Erro de Navegação Erro de Interface Erro do banco de dados Erro de Shell Script Erro de exceção Lógica Incorreta no código Tipo de dados incorreto	
Habilidade requerida para ser operada	Conhecimento sobre modelagem de sistemas orientados a objetos	
Critério de Geração dos Testes	Fluxo de controle: busca em grafo nos diagramas de atividades e de seqüência ou colaboração Fluxo de dados: a partir de um oráculo de teste	
Modelo comportamental/estrutural	Diagrama + Descrição de Caso de Uso Diagrama de Atividades Diagrama de Seqüência Diagrama de Colaboração Diagrama de Classes Dicionário de Dados em OCL	-
Tecnologia usada para modelagem	UML e OCL	-
Tipo de Técnica de Teste	Funcional	-
Entradas Requeridas	Modelo comportamental	-
Existência de um verificador de modelo	Sim	-
Existência de um mecanismo de rastreabilidade	Não	-
Compreensibilidade	-	Utiliza os conceitos padrão da UML e é apoiada por uma ferramenta
Dependências	-	Não depende de outra TTBM

AP B.2.3 Técnica 3: Chang *et al*. (1996)

CHANG, J.; RICHARDSON, D. J.; SANKAR, S. (1996), “Structural specification-based testing with ADL”, Proceedings of the 1996 International Symposium on Software Testing and Analysis ISSTA, , ACM Press, 1996, 21.

Atributo de Caracterização	**Valores**	
	Porantim	**Esquema**
Plataforma de execução do software	Independente de plataforma de execução	
Paradigma de desenvolvimento de software	Baseado em especificações formais	
Linguagem de programação	Linguagem C	
Nível(is) de Teste	Teste de Unidade	
Características de qualidade do software que podem ser avaliadas pela abordagem	Funcionalidade Segurança	
Ferramenta de Apoio	ADL Scope	
Custo associado à ferramenta	Gratuita	
Plataforma em que a ferramenta opera	Multi-plataforma	
Nível de Automação	Necessidade de modelagem intermediária	
Complexidade dos passos não-automatizados	Os dados de teste e condições booleanas precisam ser modelados durante o uso da abordagem	
Resultados gerados	Roteiros de Teste	
Tipos de Falhas que permite revelar	Erro de interface Erro de arquivo de dados Erro de Sheel Script Erro de Exceção Tipo de dados incorretos	
Habilidade requerida para ser operada	Conhecimento sobre a linguagem ADL	
Critério de Geração dos Testes	Cobertura de declaração e ramos (grafo)	
Modelo comportamental/estrutural	Especificação em ADL e Descrição dos Dados de Teste (TDD)	-
Tecnologia usada para modelagem	ADL	-
Tipo de Técnica de Teste	Funcional	-
Entradas Requeridas	Especificação em ADL e Descrição dos Dados de Teste (TDD)	-
Existência de um verificador de modelo	Não	-
Existência de um mecanismo de rastreabilidade	Sim	-
Compreensibilidade	-	Utiliza a linguagem ADL e requer modelagem de condições booleanas
Dependências	-	Não depende de outra TTBM

AP B.2.4 Técnica 4: Hartmann e Nagin (2000)

HARTMANN, A., NAGIN, K. (2004), "The AGEDIS tools for model based testing". SIGSOFT Softw. Eng. Notes 29, 4 (Jul. 2004), pp. 129-132. DOI= http://doi.acm.org/10.1145/1013886.1007529.

Atributo de Caracterização	Valores	
	Porantim	**Esquema**
Plataforma de execução do software	Sistemas Distribuídos	
Paradigma de desenvolvimento de swe	Orientado a objetos	
Linguagem de programação	Independente de linguagem de programação	
Nível(is) de Teste	Teste de Sistema	
Características de qualidade do software que podem ser avaliadas pela abordagem	Funcionalidade Segurança	
Ferramenta de Apoio	AGEDIS	
Custo associado à ferramenta	Gratuita	
Plataforma em que a ferramenta opera	Multi-plataforma	
Nível de Automação	Todos os passos são automatizados	
Complexidade dos passos não-automatizados	Simples, apenas a modelagem inicial é requerida	
Resultados gerados	Cenários de teste abstratos composto por casos de teste que cobrem às diretivas de geração de teste definidas	
Tipos de Falhas que permite revelar	Código faltando Erro de Navegação Erro de Interface Erro de arquivo de dados Erro do banco de dados Erro de Shell Script Erro de exceção Lógica Incorreta no código	
Habilidade requerida para ser operada	Conhecimento sobre modelagem de sistemas orientados a objetos e XML	
Critério de Geração dos Testes	Existe um modelo de cobertura funcional para avaliar os métodos e atributos dos objetos o software	
Modelo comportamental/estrutural	Modelo Comportamental: combinação de diagramas de classe, estado e objetos Diretivas de Geração dos Testes: Diagrama de Estado Diretivas de Execução dos Testes: arquivo XML	-
Tecnologia usada para modelagem	UML e XML	-
Tipo de Técnica de Teste	Funcional	-
Entradas Requeridas	O modelo comportamental, as diretivas de geração e execução dos testes	-
Existência de um verificador de modelo	Não	-
Existência de mecanismo de rastreabilidade	Não	-
Compreensibilidade	-	Utiliza modelos no formato padrão da UML e depende das diretivas de geração/execução
Dependências	-	Não depende de outra TTBM

AP B.2.5 Técnica 5: Kim *et al*. (1999)

KIM, Y.; HONG, H.; BAE, D.; CHA, S. (1999), “Test cases generation from UML state diagrams”, IEE Proceedings, 146, pp. 187-192.

Atributo de Caracterização	Valores	
	Porantim	**Esquema**
Plataforma de execução do software	Independente de plataforma de execução	
Paradigma de desenvolvimento de software	Orientado a objeto	
Linguagem de programação	Independente de linguagem de programação	
Nível(is) de Teste	Teste de Unidade	
Características de qualidade do software que podem ser avaliadas pela abordagem	Funcionalidade	
Ferramenta de Apoio	Não definida	
Custo associado à ferramenta	Não definida	
Plataforma em que a ferramenta opera	Não definida	
Nível de Automação	Todos os passos são automatizados	
Complexidade dos passos não-automatizados	Simples, apenas a modelagem inicial é requerida	
Resultados gerados	Casos de teste	
Tipos de Falhas que permite revelar	Erro de Navegação Erro de Interface Erro do banco de dados Erro de exceção Lógica Incorreta no código Tipo de dados incorreto	
Habilidade requerida para ser operada	Conhecimento sobre modelagem de sistemas orientados a objetos	
Critério de Geração dos Testes	Fluxo de Controle: cobertura por caminho, estado ou transação Fluxo de Dados: coberturas todas-definições, todos-usos, todos-caminhos definidos	
Modelo comportamental/estrutural	Diagrama de Estado Máquina de Estado Finito	-
Tecnologia usada para modelagem	UML Máquina de estado finito	-
Tipo de Técnica de Teste	Estrutural	-
Entradas Requeridas	Diagrama de Estado	-
Existência de um verificador de modelo	Não	-
Existência de um mecanismo de rastreabilidade	Não	-
Compreensibilidade	-	Utiliza modelo padrão da UML, o que simplifica seu entendimento
Dependências	-	Não depende de outra TTBM

AP B.2.6 Técnica 6: Friedman *et al*. (2002)

FRIEDMAN, G.; HARTMAN, A.; NAGIN, K.; SHIRAN, T. (2002), “Projected state machine coverage for software testing”, Proceedings of the 2002 ACM SIGSOFT international symposium on Software testing and analysis (ISSTA '02), ACM Press, 2002, 27.

Atributo de Caracterização	**Valores**	
	Porantim	**Esquema**
Plataforma de execução do software	Sistema concorrente	
Paradigma de desenvolvimento de software	Baseado em máquina de estado finito	
Linguagem de programação	Independente de linguagem de programação	
Nível(is) de Teste	Teste de unidade	
Características de qualidade do software que podem ser avaliadas pela abordagem	Funcionalidade Eficiência (Tempo de Resposta)	
Ferramenta de Apoio	GOTCHA	
Custo associado à ferramenta	Gratuita	
Plataforma em que a ferramenta opera	Multi-plataforma	
Nível de Automação	Todos os passos são automatizados	
Complexidade dos passos não-automatizados	Simples, apenas a modelagem inicial e pequenas escolhas são requeridas	
Resultados gerados	Casos de Teste	
Tipos de Falhas que permite revelar	Erro de Navegação Erro de Interface Erro de arquivo de dados Erro de Shell Script Erro de exceção	
Habilidade requerida para ser operada	Conhecimento sobre máquina de estado finito e sobre a linguagem Murφ	
Critério de Geração dos Testes	Cobertura de estados e transações	
Modelo comportamental/estrutural	Máquina de Estado Finito	-
Tecnologia usada para modelagem	Linguagem de descrição Murφ	-
Tipo de Técnica de Teste	Estrutural	-
Entradas Requeridas	Modelagem do sistema e restrições de teste em Murφ	-
Existência de um verificador de modelo	Sim	-
Existência de um mecanismo de rastreabilidade	Não	-
Compreensibilidade	-	Utiliza uma notação e linguagem de modelagem pouco conhecida
Dependências	-	Não depende de outra TTBM

AP B.2.7 Técnica 7: Alagar *et al*. (2000)

ALAGAR, V.; ORMANDJIEVA, O.; ZHENG, M. (2000), 'Specification-based testing for real-time reactive systems", Proceedings. 34th International Conference on Technology of Object-Oriented Languages and Systems (TOOLS), pp. 25-36.

Atributo de Caracterização	**Valores**	
	Porantim	**Esquema**
Plataforma de execução do software	Sistema Reativo de Tempo Real	
Paradigma de desenvolvimento de software	Baseado em especificação formal	
Linguagem de programação	ADA	
Nível(is) de Teste	Teste de Unidade	
Características de qualidade do software que podem ser avaliadas pela abordagem	Funcionalidade Eficiência (Tempo de Resposta)	
Ferramenta de Apoio	TROMLAB	
Custo associado à ferramenta	Gratuita	
Plataforma em que a ferramenta opera	Multi-plataforma	
Nível de Automação	Todos os passos são automatizados	
Complexidade dos passos não-automatizados	Simples, apenas a modelagem inicial é requerida	
Resultados gerados	Casos de Teste	
Tipos de Falhas que permite revelar	Erro de arquivo de dados Erro de Shell Script Erro de exceção Lógica Incorreta no código Tipo de dados incorreto	
Habilidade requerida para ser operada	Conhecimento sobre ADA e TROM	
Critério de Geração dos Testes	Cobertura de Estado e de Transação	
Modelo comportamental/estrutural	TROM (Timed Reactive Object Model)	-
Tecnologia usada para modelagem	TROM (Timed Reactive Object Model)	-
Tipo de Técnica de Teste	Funcional	-
Entradas Requeridas	Especificação do software em TROM	-
Existência de um verificador de modelo	SIM	-
Existência de um mecanismo de rastreabilidade	NÃO	-
Compreensibilidade	-	Utiliza uma linguagem de modelagem pouco conhecida
Dependências	-	Não depende de outra TTBM

AP B.2.8 Técnica 8: Stobie (2005)

STOBIE, K. (2005), “Model Based Testing in Practice at Microsoft”, In: Electronic Notes in Theoretical Computer Science, vol. 111, pp. 5 – 12.

Atributo de Caracterização	**Valores**	
	Porantim	**Esquema**
Plataforma de execução do software	Independente de plataforma de execução	
Paradigma de desenvolvimento de software	Baseado em máquina de estado finito	
Linguagem de programação	Independente de linguagem de programação	
Nível(is) de Teste	Teste de Sistema	
Características de qualidade do software que podem ser avaliadas pela abordagem	Funcionalidade	
Ferramenta de Apoio	TMT and AsmL Test Tool	
Custo associado à ferramenta	Gratuita, porém é um protótipo	
Plataforma em que a ferramenta opera	Multi-plataforma	
Nível de Automação	Todos os passos são automatizados	
Complexidade dos passos não-automatizados	Simples, apenas a modelagem inicial é requerida	
Resultados gerados	Casos de Teste em XML	
Tipos de Falhas que permite revelar	Erro de Shell Script Erro de exceção Tipo de dados incorreto	
Habilidade requerida para ser operada	Conhecimento sobre ASML	
Critério de Geração dos Testes	Busca em grafo	
Modelo comportamental/estrutural	Máquina de Estado Finito em Linguagem Abstrata de Máquina de Estado (ASML)	-
Tecnologia usada para modelagem	ASML, XML	-
Tipo de Técnica de Teste	Funcional	-
Entradas Requeridas	Especificação em ASML	-
Existência de um verificador de modelo	Não	-
Existência de um mecanismo de rastreabilidade	Não	-
Compreensibilidade	-	Utiliza uma linguagem de modelagem pouco conhecida
Dependências	-	Não depende de outra TTBM

AP B.2.9 Técnica 9: Vieira *et al*. (2006)

VIEIRA, M.; LEDUC, J.; HASLING, B.; SUBRAMANYAN, R.; KAZMEIER, J. (2006), "Automation of GUI testing using a model-driven approach", Proceedings of the 2006 international workshop on Automation of software test (AST '06), ACM Press, 2006.

Atributo de Caracterização	**Valores**	
	Porantim	**Esquema**
Plataforma de execução do software	Independente de plataforma de execução	
Paradigma de desenvolvimento de software	Orientado a objetos	
Linguagem de programação	Independente de linguagem de programação	
Nível(is) de Teste	Teste de Sistema	
Características de qualidade do software que podem ser avaliadas pela abordagem	Funcionalidade Eficiência (Tempo de Resposta)	
Ferramenta de Apoio	TDE	
Custo associado à ferramenta	Proprietária da SIEMENS	
Plataforma em que a ferramenta opera	Multi-plataforma (plugin do Eclipse)	
Nível de Automação	Todos os passos são automatizados	
Complexidade dos passos não-automatizados	Simples, apenas a modelagem inicial é requerida	
Resultados gerados	Roteiros de teste no formato TSL	
Tipos de Falhas que permite revelar	Código faltando Erro de Navegação Erro de Interface Erro de arquivo de dados Erro do ambiente Erro do banco de dados Erro de Shell Script Erro de exceção Lógica Incorreta no código Tipo de dados incorreto	
Habilidade requerida para ser operada	Conhecimento sobre modelagem de sistemas orientados a objetos e da extensão da UML usada pela abordagem	
Critério de Geração dos Testes	Fluxo de Controle: busca em grafo Fluxo de Dados: Particionamento por Categoria	
Modelo comportamental/estrutural	Diagramas de Casos de Uso, Atividade, de Classes (estendido para representar as categorias de entrada)	-
Tecnologia usada para modelagem	UML	-
Tipo de Técnica de Teste	Funcional	-
Entradas Requeridas	Diagramas UML	-
Existência de um verificador de modelo	Não	-
Existência de um mecanismo de rastreabilidade	Não	-
Compreensibilidade	-	Utiliza o padrão da UML com pequenas alterações facilmente interpretadas
Dependências	-	Não depende de outra TTBM

AP B.2.10 Técnica 10: Du Bousquet *et al*. (1999)

BOUSQUET, L. d., ZUANON, N. (1999), “An Overview of Lutess: A Specification-Based Tool for Testing Synchronous Software”. Proceedings of the 14th IEEE international Conference on Automated Software Engineering (ASE), p. 208.

Atributo de Caracterização	**Valores**	
	Porantim	**Esquema**
Plataforma de execução do software	Sistema Reativo Síncrono	
Paradigma de desenvolvimento de software	Baseado em especificação formal	
Linguagem de programação	Independente de linguagem de programação	
Nível(is) de Teste	Teste de Integração	
Características de qualidade do software que podem ser avaliadas pela abordagem	Funcionalidade Segurança	
Ferramenta de Apoio	Luteness	
Custo associado à ferramenta	Gratuita	
Plataforma em que a ferramenta opera	Multi-plataforma	
Nível de Automação	Todos os passos são automatizados	
Complexidade dos passos não-automatizados	Simples, apenas a modelagem inicial e pequenas escolhas são requeridas	
Resultados gerados	Seqüências de Teste	
Tipos de Falhas que permite revelar	Erro de Navegação Erro de Interface Erro de exceção	
Habilidade requerida para ser operada	Conhecimento sobre a linguagem Lustre	
Critério de Geração dos Testes	Dados são gerados a partir das restrições definidas na descrição do ambiente	
Modelo comportamental/estrutural	Descrição do Ambiente	-
Tecnologia usada para modelagem	Lustre	-
Tipo de Técnica de Teste	Funcional	-
Entradas Requeridas	Descrição do Ambiente e um oráculo de teste	-
Existência de um verificador de modelo	Sim	-
Existência de um mecanismo de rastreabilidade	Não	-
Compreensibilidade	-	Utiliza uma linguagem de modelagem pouco conhecida
Dependências	-	Não depende de outra TTBM

AP B.2.11 Técnica 11: Chung *et al*. (1999)

CHUNG, I. S., KIM, H. S., BAE, H. S., KWON, Y. R., LEE, B. S. (1999), “Testing of Concurrent Programs Based on Message Sequence Charts”, Proceedings of the international Symposium on Software Engineering For Parallel and Distributed Systems (PDSE), pp. 72.

Atributo de Caracterização	**Valores**	
	Porantim	**Esquema**
Plataforma de execução do software	Sistema Concorrente	
Paradigma de desenvolvimento de software	Baseado em Especificação Formal	
Linguagem de programação	Independente de linguagem de programação	
Nível(is) de Teste	Teste de Unidade	
Características de qualidade do software que podem ser avaliadas pela abordagem	Funcionalidade	
Ferramenta de Apoio	Não definido	
Custo associado à ferramenta	Não definido	
Plataforma em que a ferramenta opera	Não definido	
Nível de Automação	Médio – alguns passos manuais são requeridos	
Complexidade dos passos não-automatizados	Requer modelagem formal da aplicação usando modelos determinísticos e não determinísticos e definição de restrições	
Resultados gerados	Casos de teste	
Tipos de Falhas que permite revelar	Erro de arquivo de dados Erro de Shell Script Erro de exceção	
Habilidade requerida para ser operada	Conhecimento sobre Message Sequence Charts (MSC)	
Critério de Geração dos Testes	Busca em grafo no modelo MSC	
Modelo comportamental/estrutural	MSC	-
Tecnologia usada para modelagem	MSC	-
Tipo de Técnica de Teste	Estrutural	-
Entradas Requeridas	MSC	-
Existência de um verificador de modelo	Não	-
Existência de um mecanismo de rastreabilidade	Não	-
Compreensibilidade	-	Requer conhecimento sobre MSC, utilizado constantemente em sistemas de telecomunicação
Dependências	-	Não depende de outra TTBM

AP B.2.12 Técnica 12: Parissis e Vassey (2003)

PARISSIS, I., VASSEY, J. (2003), “Thoroughness of Specification-Based Testing of Synchronous Programs”. Proceedings of the 14th international Symposium on Software Reliability Engineering (ISSRE), Washington, DC, p. 191.

Atributo de Caracterização	**Valores**	
	Porantim	**Esquema**
Plataforma de execução do software	Sistema Reativo Síncrono	
Paradigma de desenvolvimento de software	Baseado em especificação formal	
Linguagem de programação	Independente de linguagem de programação	
Nível(is) de Teste	Teste de Sistema	
Características de qualidade do software que podem ser avaliadas pela abordagem	Funcionalidade Segurança	
Ferramenta de Apoio	Luteness	
Custo associado à ferramenta	Gratuita	
Plataforma em que a ferramenta opera	Multi-plataforma	
Nível de Automação	Todos os passos são automatizados	
Complexidade dos passos não-automatizados	Simples, apenas a modelagem inicial e pequenas escolhas são requeridas	
Resultados gerados	Casos de teste	
Tipos de Falhas que permite revelar	Erro de Navegação Erro de arquivo de dados Erro de Shell Script Erro de exceção	
Habilidade requerida para ser operada	Conhecimento sobre a linguagem LUSTRE	
Critério de Geração dos Testes	Cobertura por estados	
Modelo comportamental/estrutural	Descrição do ambiente em LUSTRE	-
Tecnologia usada para modelagem	LUSTRE	-
Tipo de Técnica de Teste	Funcional	-
Entradas Requeridas	Descrição do ambiente em LUSTRE e um oráculo	-
Existência de um verificador de modelo	Não	-
Existência de um mecanismo de rastreabilidade	Não	-
Compreensibilidade	-	Utiliza uma linguagem formal para modelagem do software sob teste
Dependências	-	Não depende de outra TTBM

AP B.2.13 Técnica 13: Dalal *et al*. (1999)

DALAL, S.; JAIN, A.; KARUNANITHI, N.; LEATON, J. M.; Lott, C. M.; Patton, G. C.; Horowitz, B. M. (1999), “Model-based testing in practice”, In: ICSE’99, May, pp. 285--294.

Atributo de Caracterização	**Valores**	
	Porantim	**Esquema**
Plataforma de execução do software	Independente de plataforma de execução	
Paradigma de desenvolvimento de software	Baseado em especificação formal	
Linguagem de programação	Independente de linguagem de programação	
Nível(is) de Teste	Teste de Sistema	
Características de qualidade do software que podem ser avaliadas pela abordagem	Funcionalidade Segurança	
Ferramenta de Apoio	AETG	
Custo associado à ferramenta	Gratuita	
Plataforma em que a ferramenta opera	Não especificado	
Nível de Automação	Todos os passos são automatizados	
Complexidade dos passos não-automatizados	Simples, apenas a modelagem inicial é requerida	
Resultados gerados	Casos de Teste prontos para serem executados	
Tipos de Falhas que permite revelar	Erro de Navegação Erro de Interface Erro do banco de dados Erro de exceção Lógica Incorreta no código Tipo de dados incorreto	
Habilidade requerida para ser operada	Conhecimento sobre a linguagem de modelagem AETGSpec	
Critério de Geração dos Testes	Classe de Equivalência	
Modelo comportamental/estrutural	AETGSpec	-
Tecnologia usada para modelagem	AETGSpec	-
Tipo de Técnica de Teste	Funcional	-
Entradas Requeridas	Modelagem de dados de teste	-
Existência de um verificador de modelo	Não	-
Existência de um mecanismo de rastreabilidade	Não	-
Compreensibilidade	-	Média (Dependente da linguagem adotada)
Dependências	-	Não depende de nenhuma TTBM

APÊNDICE C – PLANO DO ESTUDO DE AVALIAÇÃO DE *PORANTIM* + APOIO FERRAMENTAL PROVIDO POR MARAKÁ

Este apêndice descreve os instrumentos usados no estudo de avaliação da infra-estrutura computacional de apoio à abordagem Porantim.

AP C.1 Instrumentos

AP C.1.1 Formulário E0 – Caracterização de Participantes

Formulário E0 – Caracterização do Participante

Eu declaro que concordo em participar em estudos não invasivos conduzidos pelo Prof. Guilherme Horta Travassos e pesquisador Arilo Claudio Dias Neto. Estes estudos visam compreender a viabilidade de aplicação de abordagens e ferramentas de apoio à seleção de técnicas de teste de software baseado em modelos em projetos de software.

Assinatura: _____

Nome: _____ **Data:** _____

Objetivo: conhecer o histórico, conhecimento e experiência do participante

Formação do participante

1. Qual o seu grau de formação mais alto em Engenharia de Software?

   ( ) Graduação ( ) Especialização ( ) Mestrado ( ) Doutorado

Experiência no Desenvolvimento de Software

2. Há quanto tempo você estima que atua no desenvolvimento de software? _____
3. Quantos projetos de software você estima que tenha participado? _____

Experiência em Teste de Software

4. Como você avalia seu grau de conhecimento em teste de software?

   ( ) Baixo ( ) Médio ( ) Alto ( ) Excelente

5. Há quanto tempo você estima que trabalha com a realização de teste de software? _____
6. Você já teve a experiência de selecionar técnicas de teste para um projeto de software? ( ) Sim ( ) Não

Experiência em Teste de Software Baseado em Modelos

7. Como você avalia seu grau de conhecimento em teste baseado em modelos?

   ( ) Baixo ( ) Médio ( ) Alto ( ) Excelente

8. Há quanto tempo você estima que trabalha com teste de software baseado em modelos? _____

9. Você já teve a experiência de selecionar técnicas de teste baseado em modelos para um projeto de software? ( ) Sim ( ) Não

Experiência na Seleção de Técnicas de Teste Baseado em Modelos

10. Se você fosse encarregado de selecionar técnicas de teste baseado em modelos para um projeto de software, qual estratégia você adotaria para escolher 1 única técnica?

_____

_____

11. Se você fosse encarregado de selecionar técnicas de teste baseado em modelos para um projeto de software, qual estratégia você adotaria para escolher mais de 1 técnica para serem combinadas em um projeto de software?

_____

_____

AP C.1.2 Formulário E1 – Execução do Exercício sem apoio ferramental

Formulário E1 – Seleção sem apoio ferramental

Nome: _____ **Data:** _____

Objetivo: Medir os tempos e indicar as TTBMs descartada e selecionadas

Exercício

Os passos a serem seguidos são:

1. Para realizar este exercício relembre que você deve:
a. Ler a documentação do projeto.
b. Examinar uma a uma as técnicas de TBM disponíveis.
c. Descartar 1 TTBM que você julga inadequada ao projeto.
d. Decidir quais são as 2 TTBMs que você usaria para o projeto.

(ETAPA 1)

2. Informe o período em minutos que você gastou para analisar a documentação do projeto e as técnicas de TBM para escolher qual será descartada: _____ (minutos)

3. Indique o ID da TTBM que você descartaria para seleção neste projeto: _____

Justifique sua resposta:

_____

_____

(ETAPA 2)

4. Informe o período em minutos que você gastou para analisar a documentação do projeto e as TTBMs para escolher quais seriam as 2 TTBMs selecionadas para o projeto: _____

5. Indique qual o ID das TTBMs que você combinaria para este projeto:

a. Técnica de TBM 1: _____

b. Técnica de TBM 2: _____

Justifique sua resposta:

_____

_____

AP C.1.3 Formulário E2 – Execução do Exercício com o apoio ferramental provido por *Porantim*

Formulário E2 – Seleção com apoio ferramental de *Porantim*

Nome: _____ **Data:** _____

Objetivo: Medir os tempos e indicar as técnicas de TBM descartada e selecionadas

Exercício

Os passos a serem seguidos são:

1. Para realizar este exercício relembre que você deve:

a. Ler a documentação do projeto.

b. Preencher a caracterização do projeto na ferramenta e salvá-la

c. Examinar uma a uma as TTBMs sugeridas pela ferramenta e decidir quais são as 2 TTBMs que você selecionaria para o projeto.

d. Analisar (sem a ferramenta) as 5 TTBMs e indicar qual delas você descartaria por julgar inadequada ao projeto.

(ETAPA 1)

2. Informe o período em minutos que você gastou para analisar a documentação do projeto e preencher a sua caracterização na ferramenta: _____ (minutos)

(ETAPA 2)

3. Informe o período em minutos que você gastou para analisar a documentação do projeto e as TTBMs sugeridas pela ferramenta para escolher quais seriam as 2 TTBMs selecionadas para o projeto: _____

4. Indique qual o ID das TTBMs que você selecionaria de forma combinada para este projeto:

a. Técnica de TBM 1: _____

b. Técnica de TBM 2: _____

(ETAPA 3)

5. Informe o período em minutos que você gastou para analisar a documentação do projeto e das 5 TTBMs (sem o apoio ferramental) para indicar para qual seria descartada por você: _____ (minutos)

6. Indique qual o ID da TTBM que você descartaria para seleção neste projeto: _____

Justifique sua resposta:

_____

_____

AP C.1.4 Formulário E3 – Avaliação do apoio ferramental provido por *Porantim*

Formulário E3 – Avaliação do apoio ferramental provido por *Porantim*

Nome: _____ **Data:** _____

Objetivo: Verificar o potencial que os participantes observam sobre o apoio ferramental construído para *Porantim*

Avaliação Geral do Apoio Ferramental

1. Quais os pontos positivos e negativos relacionados ao uso do apoio ferramental provido por *Porantim*?

- NEGATIVO: _____
- POSITIVO: _____

2. Quais as sugestões de melhorias que você poderia relatar no que diz respeito ao processo de seleção de técnicas de TBM para projetos de software?

_____

3. Como você avalia o grau de dificuldade em usar *Porantim* e seu apoio ferramental comparada à seleção manual (sem apoio ferramental)?

( ) Muito Baixo ( ) Baixo ( ) Médio ( ) Alto

Avaliação do PASSO 1: Caracterização do Projeto de Software

4. Existe alguma sugestão de melhoria para o passo 1 do processo de seleção de TTBMs (Caracterização do Projeto de Software)?

_____

_____

Avaliação do PASSO 2: Seleção de Técnicas de TBM

5. Como você avalia os resultados providos pela funcionalidade de *Porantim* que calcula o grau de adequação e ordena as TTBMs por este indicador?

( ) Úteis ( ) Adequados ( ) Confusos ( ) Não-Adequados

6. A exibição da adequação entre uma técnica de TBM e o projeto de software através do gráfico de radar ajudou você no processo de seleção das TTBMs?

( ) Sim ( ) Pouco ( ) Não Ajudou ( ) Dificultou

Justifique sua resposta:

_____

_____

Avaliação do PASSO 3: Análise da Combinação de Técnicas de TBM

7. Como você avalia os resultados providos pela funcionalidade de *Porantim* que analisa a combinação de TTBMs e provê indicadores do impacto desta combinação no processo de testes?

( ) Úteis ( ) Adequados ( ) Confusos ( ) Não-Adequados

8. A exibição do impacto da combinação de TTBMs no processo de teste por meio de um gráfico de Gauge ajudou você no processo de seleção das TTBMs?

( ) Sim ( ) Pouco ( ) Não Ajudou ( ) Dificultou

Justifique sua resposta:

_____

_____