UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO

FERRAMENTA DE APOIO À IMPLEMENTAÇÃO DO PROCESSO MELHORIA DE PROCESSO DE TESTE (MPT.BR)

VANDER BERTOLINI

BLUMENAU
2014

2014/1-24

VANDER BERTOLINI

FERRAMENTA DE APOIO À IMPLEMENTAÇÃO DO PROCESSO MELHORIA DE PROCESSO DE TESTE (MPT.BR)

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Ciência da Computação — Bacharelado.

Prof. Jacques Robert Heckmann, Mestre - Orientador

BLUMENAU
2014

2014/1-24

FERRAMENTA DE APOIO À IMPLEMENTAÇÃO DO PROCESSO MELHORIA DE PROCESSO DE TESTE (MPT.BR)

Por

VANDER BERTOLINI

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

Presidente:	Prof. Jacques Robert Heckmann, Mestre – Orientador, FURB
Membro:	Prof. Marcel Hugo, Mestre – FURB
Membro:	Prof. Alexander Roberto Valdameri, Mestre – FURB

Blumenau, 10 de julho de 2014

Dedico este trabalho à minha esposa que me apoiou em todos os momentos para que a realização deste fosse possível.

AGRADECIMENTOS

A Deus, pela força concedida ao enfrentar os obstáculos da vida.

À minha família, que sempre me mostrou a importância do estudo.

Aos meus amigos, em especial para dois que tiveram grande importância na realização deste trabalho. O primeiro Deivid Sant´ana que desde o início da realização deste trabalho procurou da melhor forma me apoiar nas minhas escolhas, dando conselhos e me mostrando caminhos que seriam mais fáceis de seguir. Com seu jeito brincalhão fez valer a pena os anos que passei na FURB. Muitas vezes eu sabia que a aula ia ser matada, mas mesmo assim a vontade de ir para a universidade era enorme porque sabia que passaria algumas horas ao lado de um grande amigo. O segundo José Guilherme Vanz mais conhecido como “Milico”, esse cara realmente merece aplausos, com seu jeito simples e tranquilo, soube me ajudar a sanar os problemas que tive na realização do TCC e foi de fundamental importância para que eu conseguisse alcançar os objetivos propostos para este trabalho.

Ao meu orientador, Jacques Robert Heckmann, por ter acreditado e me apoiado no desenvolvimento deste trabalho.

Às vezes você só percebe a importância de um momento quando ele se torna uma lembrança.

Sharpie Thoughts

RESUMO

O presente trabalho é um estudo sobre o modelo de processo de testes Melhoria de Processo de Teste Brasileiro (MPT.BR). Aborda uma alternativa para pequenas e médias empresas que procuram um custo mais acessível na implementação de um processo. Este trabalho é uma continuação do trabalho de conclusão de curso de Daniel Ricardo de Amorim (AMORIM, 2011), que abordou os dois primeiros dos cinco níveis de maturidade do MPT.BR. Neste trabalho é abordado adicionalmente o nível número três. Por fim, é desenvolvida uma ferramenta, em forma de *plug-in* para o Eclipse, para ajudar na gerência de projetos de teste aderentes ao MPT.BR versão 3.1, através da automatização de algumas áreas dos três níveis do processo pesquisado.

Palavras-chave: MPT.BR. Gerência de projetos de teste. *Plug-in.*

ABSTRACT

The present work is a study based on the tests process model Improvement of Brazilian Test Procedure (MPT.BR). It discusses an alternative for small and medium companies which look for an accessible process implementation cost. This work is a Daniel Ricardo Amorim's conclusion work continuation, in which he addressed the first two between five levels of maturity of the MPT.BR. In this work, the level three is also addressed. Finally, a tool is developed in the form of an Eclipse plug-in, to help test project management adherent to MPT.BR version 3.1, by automating some areas of the level three of the researched process.

Key-words: MPT.BR. Project management test. Plug-in.

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

API – *Application Programming Interface*

CMMI – Modelo de Maturidade em Capacitação Integração

FDT – Fechamento do Teste

GPT – Gerência de Projetos de Teste

GRT – Gerência de Requisitos de Teste

IEEE – *Institute of Electrical and Electronic Engineers*

JDBC – *Java Database Connectivity*

JDT – *Java Development Tools*

MAT – Medição e Análise do Teste

MER – Modelo Entidade Relacionamento

MPT.BR – Melhoria do Processo de Teste

OGT – Organização do Teste

PDE – *Plug-in Development Environment*

PDF – *Portable Document Format*

PET – Projeto e Execução de Teste

SDK – *Software Development Kit*

SWT – *Standard Widget Toolkit*

TDA – Teste de Aceitação

SUMÁRIO

1 INTRODUÇÃO

Diversas empresas optam pela adoção de processos que contribuam no gerenciamento de mudanças. O objetivo destes processos é assegurar que os projetos desenvolvidos sejam fáceis de ser adaptados ou melhorados (OLIVEIRA 2006, p. 14). Com a crescente expansão do mercado e clientes cada vez mais exigentes, ter um modelo de processo que possibilite a análise do projeto acaba sendo um diferencial, principalmente com a concorrência presente nos dias de hoje.

> A busca por modelos está diretamente vinculada à demanda organizacional, visto que a efetiva gestão dos ativos organizacionais é crítica para o sucesso do negócio. Nesse contexto, os processos, oriundos de modelos de maturidade, têm por objetivo auxiliar as organizações a alcançarem os resultados almejados através da melhor execução das atividades planejadas e também minimizar os impactos quando da introdução e uso de novas tecnologias. (SOFTEX RECIFE, 2013b).

Segundo Pezzè e Young (2008, p. 397), qualquer processo complexo requer planejamento e monitoração. O processo de qualidade requer a coordenação de muitas atividades diferentes por um período equivalente a um ciclo completo de desenvolvimento. Para ordenar, provisionar e coordenar todas as atividades que apoiam o objetivo de qualidade é preciso planejar. A monitoração do estado real frente ao planejado se faz necessária para orientar o ajuste do processo.

Segundo Bastos et al. (2007, p. 18), "Defeitos encontrados durante a produção tendem a custar muito mais que defeitos encontrados em modelos de dados e em outros documentos dos projetos do software".

Sendo o MPT.BR um modelo novo, não há no mercado um software que atenda por completo as práticas exigidas pelo modelo. Desta forma, as empresas que adotam este modelo investem em diversas ferramentas e em treinamento dos usuários para seu uso, cada uma abrangendo algumas práticas do modelo. Com todas estas ferramentas são feitos gastos com a compra e treinamento dos usuários.

Diante do exposto, dar-se-á continuidade ao trabalho de conclusão de curso de Amorim (2011), intitulado "Ferramenta de Apoio à Implementação do Processo Melhoria de Processo de Teste (MPT)", cujo objetivo principal consistiu em desenvolver um *plug-in* para Eclipse, para apoiar o desenvolvimento do modelo. Este trabalho anterior atendia às práticas genéricas e específicas do nível 1 e 2 de um total de 5 níveis de maturidade estabelecidas pelo modelo MPT.BR.

No presente trabalho desenvolver-se-á uma ferramenta que contemple também o nível 3 do modelo MPT.BR, chamado de "definido", e agregue os níveis 1 e 2 do trabalho anterior, para automatizar o modelo, sob a forma de um *plug-in* para o Eclipse. Isso facilitará o

trabalho de adoção do modelo MPT.BR das pequenas e médias empresas, ajudando-as a tornar seus produtos cada vez mais confiáveis.

O Eclipse servirá de suporte para o *plug-in* por ser uma ferramenta gratuíta de fácil acesso e que abrange a linguagem Java, que também é gratuita e não exige custos.

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho é desenvolver um *plug-in* de apoio à implementação do processo melhoria de processo de teste (MPT.BR).

Os objetivos específicos do trabalho são:

a) atualizar os níveis de maturidade 1 e 2 desenvolvidos por Amorim (2011), baseados na versão 2.0 do MPT.BR para a sua versão 3.1;
b) prover a ferramenta com recursos para a gerência de projetos de teste, de forma que atenda ao nível 3 do MPT.BR, abordando as áreas de processo de fechamento do teste, medição e análise de teste, teste de aceitação, gerência de projetos de teste, e projeto e execução de teste.

1.2 ESTRUTURA DO TRABALHO

No segundo capítulo deste trabalho é apresentada uma revisão bibliográfica abordando os temas necessidade de um modelo de testes, para garantir a qualidade e autonomia do software a ser desenvolvido; documentação padronizada da IEEE-829, que é base para a criação dos modelos de processo de teste; o modelo MPT.BR, abordando as vantagens do seu uso; os processos necessários para receber uma certificação MPT.BR, bem como sua divisão em níveis de maturidade; e conceitos do processo que servem como base para a criação da ferramenta. É comentado o trabalho de conclusão de curso intitulado Ferramenta de Apoio a Implementação do Processo Melhoria de Processo de Teste (AMORIM, 2011), apontando uma breve descrição sobre o mesmo. Também é apresentado o ambiente *Plug-in Development Environment* (PDE), que é o utilizado para o desenvolvimento da ferramenta em forma de *plug-in* do Eclipse. Por fim, apresenta-se algumas ferramentas que possuam algumas semelhanças com o propósito deste trabalho.

No terceiro capítulo são apresentadas as funcionalidades da ferramenta desenvolvida neste trabalho, ferramentas auxiliares usadas para o desenvolvimento, implementação utilizando um estudo de caso com os resultados encontrados e os desafios encontrados na construção da ferramenta.

No quarto capítulo são mostradas as conclusões deste trabalho e sugestões para futuras continuações do mesmo.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são explorados temas como teste de software, qualidade de software, padrão de documentação, automação de processos de teste e modelo de processo de teste (MPT.BR). É feita uma abordagem ao PDE, que é o framework utilizado para a criação da ferramenta. Por fim, são comentados os trabalhos correlatos que possuem alguma relação com a ferramenta criada.

2.1 NECESSIDADE DE UM MODELO DE TESTE DE SOFTWARE

Gerenciar um processo de qualidade envolve o planejamento de um conjunto de atividades com seus custos e ajustes de qualidade, monitoração do progresso para identificar riscos o mais cedo possível, para evitar atrasos e ajustar o plano quando necessário. Estas atividades requerem criatividade humana e percepções que nenhuma ferramenta pode substituir. Apesar disso, ferramentas podem apoiar o processo de gerência, melhorando a tomada de decisão pela organização e monitoração de atividades e resultados, facilitando a interação do grupo, gerenciando documentos de qualidade e acompanhando custos (PEZZÈ; YOUNG 2008, p. 469).

Segundo Pezzè e Young (2008, p. 468), ainda que diligentes e bem-intencionados, os humanos são lentos e sujeitos a erros quando lidam com tarefas repetitivas. Por outro lado, tarefas simples e repetitivas normalmente são fáceis de automatizar, enquanto que julgamento e solução criativa de problemas permanecem fora do domínio da automação. Seres humanos são muito bons para identificar cenários relevantes de execução que correspondam a especificações de casos de teste, mas são muito ineficientes na geração de grandes volumes de casos de teste. Desta forma, havendo ferramentas que ajudem os humanos, faz com que as tarefas sejam executadas com mais confiança nos resultados, evitando erros desnecessários e que possam causar problemas mais sérios ao projeto.

2.2 PADRÃO DE DOCUMENTAÇÃO IEEE-829

Para Miguel (2013):

> O padrão IEEE-829, está relacionado com o processo de testes, etapa do processo de desenvolvimento de software de suma importância para garantia e controle da qualidade. Sua abrangência vai desde testes unitários até testes de aceitação e tem por objetivo definir documentos consistentes e adequados capazes de definir, registrar e prover condições de análise dos resultados obtidos ao longo do processo.

Gerenciar um processo de teste de software não é uma tarefa fácil. Várias vezes os seres humanos deparam-se com diversas falhas, as quais não os deixam seguir um

cronograma lógico, em virtude de um cenário crítico, na grande maioria dos casos, seguidos de cobranças e expectativas que não estão de acordo com o cenário do projeto e das condições de trabalho disponibilizadas aos profissionais (MIGUEL, 2013).

Segundo Miguel (2013), o processo de teste de software é melhorado com a produção de documentos confeccionados a partir de templates definidos, padronizados e reconhecidos. O simples fato de existir um modelo contendo informações nas quais devemos preencher com metas, indicadores ou resultados de um processo nos conduz a assertividade e evita que o esquecimento seja causador de lacunas, falhas causadas por defeitos manifestados a partir de um erro de programação. Segundo Rios (2008, p. 30), documentos padronizados facilitam a comunicação entre as partes envolvidas pois definem uma forma de comunicação em comum.

O padrão IEEE-829 apresenta um conjunto de documentos. Um destes é o plano de teste.

> O Plano de Teste, documento básico gerado através da etapa de planejamento, serve como escopo de referência durante a execução do teste e também serve como documento de comunicação junto ao cliente que contratou o serviço de teste. Ao passarmos para a etapa de especificação precisamos ter em mãos o Plano de Teste. (RIOS, 2008, p. 30).

Um plano de teste da IEEE-829 deve conter em seu padrão campos como identificador, introdução, itens de teste, funcionalidades a serem testadas, funcionalidades a não serem testadas, abordagem de teste, critérios de término dos testes, critérios de suspensão dos testes, entregas, tarefas, ambientes, responsabilidades, treinamentos, cronograma, riscos e aprovações (RIOS, 2008, p. 37).

2.3 MODELO DE PROCESSO DE TESTE (MPT.BR)

Segundo Couto (2007, p. 130), “Estudos sobre a qualidade no setor de software brasileiro mostraram a necessidade de um esforço significativo capaz de aumentar a maturidade dos processos de software das empresas brasileiras”.

> O MPT.Br é um modelo para Melhoria do Processo de Teste de Software Brasileiro, que está sendo desenvolvido com o principio básico de ser compatível com o modelo MPS.BR criado pela Softex e é baseado também em alguns critérios usados pelo CMMI. O MPT é voltado para a melhoria das áreas de teste de software de empresas de qualquer porte. O modelo é leve e passível de ser adotado por área de teste de software para apurar o seu nível de maturidade, sem com isso onerar os seus processos anteriormente implementados. O objetivo principal será garantir que as áreas de teste de software de tamanho reduzido possam ser avaliadas e estimuladas a alcançarem níveis maiores de maturidade, sem que para isso tenham que incorrer em altos custos operacionais (ASSOCIAÇÃO LATINO AMERICANA DE TESTE DE SOFTWARE, 2002).

Por possuir foco nas pequenas e micro empresas, os custos de implantação do modelo são relativamente baixos, conforme ilustrados no quadro 1.

Quadro 1 – Custos de implantação do modelo MPT.BR

	CONSULTORIA	**AVALIAÇÃO**	**TEMPO**	**VALOR DE REFERÊNCIA**
NÍVEL 1	74 HORAS	1 DIA	4 A 5 MESES	R$ 16.500,00
NÍVEL 2	64 HORAS	1,5 DIAS	4 A 5 MESES	R$ 16.500,00
NÍVEL 3	113 HORAS	3 DIAS	5 A 6 MESES	R$ 30.000,00
NÍVEL 4	102 HORAS	3,5 DIAS	5 A 6 MESES	R$ 30.000,00
NÍVEL 5	130 HORAS	4 DIAS	5 A 6 MESES	R$ 36.000,00

Fonte: Softex (2013b).

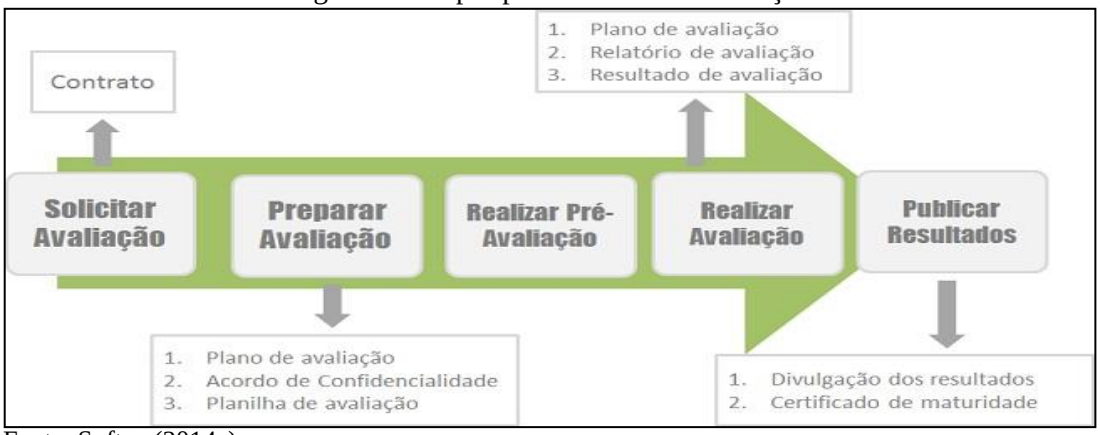
Segundo Softex (2013b), as principais diferenças entre o modelo MPT.BR e os demais são:

a) foi projetado com foco nas características e necessidades das pequenas e microempresas desenvolvedoras de software;
b) possui processos de implantação curtos, adequados às necessidades específicas de cada empresa;
c) é aderente aos principais modelos de maturidade internacionais;
d) é 100% aderente às metodologias ágeis;
e) possui baixo custo de implantação quando comparado aos principais modelos internacionais.

2.3.1 Como certificar a sua empresa com o modelo

Segundo Softex (2014c), são realizadas oito etapas para uma empresa conseguir uma certificação do MPT.BR. Na figura 1 são ilustradas as etapas para receber a certificação.

Figura 1 – Etapas para receber a certificação



Fonte: Softex (2014c).

O processo é iniciado com a identificação de uma avaliação pela organização interessada:

1. a organização entra em contato com uma instituição avaliadora que elabora uma proposta de avaliação;
2. uma vez aceita a proposta de avaliação, tem início o planejamento, que envolve a elaboração do plano de avaliação e da agenda da avaliação, seguida pela preparação da avaliação que inclui o preenchimento da planilha de evidências e assinatura do acordo de confidencialidade pelo avaliador líder e pela organização;
3. finalizando a preparação da avaliação, o avaliador realiza uma pré-avaliação para garantir que as evidências estão disponíveis e verificar se há alguma pendência para a realização da avaliação;
4. caso haja alguma questão a ser resolvida, volta-se para a preparação da avaliação;
5. se o parecer da pré-avaliação for positivo, é iniciada a avaliação, onde a equipe de avaliação coordenada pelo avaliador líder analisa as evidências, realiza entrevistas com integrantes da avaliação, identifica os achados da organização e caracterizam o nível de maturidade da organização;
6. todas as informações da avaliação são então documentadas pelo avaliador líder e pela equipe de avaliação, produzindo o relatório da avaliação e o resultado da avaliação;
7. as informações documentadas no relatório da avaliação e no resultado da avaliação são também compiladas em apresentação dos resultados, que são oferecidas à organização;

8. a avaliação, junto com a documentação elaborada, também deve ser auditada pela unidade executora. Caso questões sejam identificadas na auditoria, estas devem ser resolvidas pelo avaliador líder.

Ocorrendo os passos com sucesso, o resultado da avaliação é então publicado pela unidade executora.

2.3.2 Os cinco níveis de maturidade do modelo

Segundo Softex (2013a), o modelo MPT.BR apresenta cinco níveis de maturidade que representam a evolução do processo de teste de uma organização. Os níveis e as práticas de cada nível são representados a seguir.

2.3.2.1 Nível 1 – Parcialmente gerenciado

Este nível representa o primeiro patamar de maturidade de uma organização. Ele contém o mínimo que uma organização precisa para demonstrar que a disciplina de teste é aplicada nos projetos e que esta aplicação ocorre de forma planejada e controlada.

As áreas do processo são listadas abaixo:

a) Gerência de Projetos de Teste (GPT): é fundamental que um planejamento efetivo seja realizado para minimizar os riscos associados ao projeto, e este inclui a definição de objetivos do teste, análise de risco do produto, definição da estratégia de teste, definição do escopo do teste, estimativa de atributos de produtos de trabalho e tarefas, determinação de recursos necessários, negociação de compromissos, elaboração de um cronograma e análise de riscos do projeto. Estas informações devem ser reunidas e revisadas no Plano de Teste, documento que relaciona todos os artefatos do planejamento e representa o guia do projeto de teste (SOFTEX RECIFE, 2013a, p. 83);
b) Projeto e Execução de Teste (PET): o teste sistemático implica que casos de teste sejam identificados e elaborados, os resultados produzidos pelo sistema sejam sistematicamente comparados com os resultados esperados e as divergências sejam reportadas na forma de incidentes. Em uma organização de baixa maturidade em teste, o conjunto de práticas presentes nesta área de processo visa garantir que os testes estão sendo executados corretamente. Desta forma, para o Nível 1 de maturidade não existe ainda a preocupação com o uso adequado da documentação, mas sim a garantia que as atividades essenciais estão sendo cumpridas (SOFTEX RECIFE, 2013a, p. 125).

2.3.2.2 Nível 2 – Gerenciado

Neste segundo nível a aplicação do processo de teste na organização possui maior visibilidade. O escopo do projeto passa a ser controlado pelo processo de gestão de mudanças, padrões são definidos e os processos são monitorados e controlados.

A área do processo deste nível, além das áreas do processo do nível 1, é a Gerência de Requisitos de Teste (GRT), todos os requisitos que são fornecidos ou originados do projeto de teste devem ser gerenciados. Esta gestão implica na aprovação dos requisitos junto aos fornecedores, obtenção de um compromisso com a equipe técnica para desenvolvimento, a gestão das mudanças ocorridas nos requisitos do projeto e a identificação de inconsistências entre requisitos e produtos de trabalho do projeto. Outra parte importante da gestão aborda a manutenção da rastreabilidade bidirecional entre os requisitos e os produtos de trabalho do projeto. Esta rastreabilidade é a base para uma análise de impacto da mudança efetiva dos requisitos (SOFTEX RECIFE, 2013a, p. 103).

2.3.2.3 Nível 3 – Definido

No terceiro nível do modelo o teste se torna organizacional. Processos padrões de teste são adotados, a garantia da qualidade é instituída de modo a auxiliar a definição dos processos, são definidas responsabilidades para a organização do teste e um programa de medição é implantado na organização. Também neste nível o ciclo de vida do teste é integrado ao ciclo de vida do desenvolvimento, o teste estático e de aceitação são formalizados e procedimentos sistemáticos são aplicados para o fechamento do teste.

As áreas do processo deste nível, além das áreas do processo do nível 1, são listadas abaixo:

a) Fechamento Do Teste (FDT): envolve os procedimentos realizados no término das atividades de teste. Um exemplo típico da finalização do teste ocorre quando os critérios de saída do teste estão satisfeitos e que aquela fase ou tipo de teste podem ser concluídos (SOFTEX RECIFE, 2013a, p. 65);
b) Garantia Da Qualidade (GDQ): dá suporte à entrega de produtos de alta qualidade, fornecendo visibilidade à gerência nos processos e produtos de trabalho. As avaliações de produtos de trabalho e processo realizadas identificam a aderência destes aos procedimentos e padrões adotados, assim como auxilia na institucionalização destes procedimentos e padrões (SOFTEX RECIFE, 2013a, p. 79);

c) Medição e Análise de Teste (MAT): a gerência de uma organização possui diversas necessidades de informação referentes ao teste e os seus processos que na maioria das vezes vão ser satisfeitas através das atividades de medição e análise de teste. Sendo assim, é necessário um trabalho sistemático para identificar estas necessidades de informação relacionadas ao teste e como estas serão atendidas (SOFTEX RECIFE, 2013a, p. 107);
d) Organização do Teste (OGT): trata dos aspectos ligados à institucionalização do processo de teste considerando a formação de um grupo gestor chamado Grupo de Processo de Teste de Software (GPTS). Também é tratada a definição de processos padrões para teste, regras para sua adaptação e evolução controlada do processo de teste a partir da definição e implantação de ações de melhoria (SOFTEX RECIFE, 2013a, p. 113);
e) Teste De Aceitação (TDA): é um nível de teste formal conduzido para observar se um determinado produto de software satisfaz os seus critérios de aceitação e capacitar o cliente a determinar se ele aceita ou não aquele produto. O teste de aceitação objetiva determinar se o produto está apto para o uso (SOFTEX RECIFE, 2013a, p. 131);
f) Teste Estático (TES): é caracterizado pela não execução do software. A principal representação do teste estático é o uso de revisões, que consiste em uma avaliação sistemática de produtos de trabalho realizada por uma equipe competente para identificação de defeitos e atendimento aos requisitos de produto de trabalho (SOFTEX RECIFE, 2013a, p. 137);
g) Treinamento (TRE): envolve a identificação de necessidades de treinamento estratégico para a organização. Um programa de treinamento estratégico deve ser elaborado para satisfazer as necessidades de treinamento da organização (SOFTEX RECIFE, 2013a, p. 147).

2.3.2.4 Nível 4 - Prevenção de defeitos

O quarto nível do modelo é focado na prevenção de defeitos e melhoria sistemática da qualidade do produto. Neste nível um processo de gestão de defeitos existe na organização, onde defeitos encontrados mais cedo no ciclo de vida são acompanhados e ações proativas são tomadas para evitar que novos defeitos originados pelas mesmas causas raiz ocorram. Uma análise de risco dos atributos não-funcionais do produto e atividades de teste não-funcional

são executadas para minimizar estes riscos e também é realizada uma análise para determinar a eficácia do teste e a determinação com dados objetivos do nível de qualidade do produto.

As áreas do processo deste nível, além da área de processo do nível 3 OGT, são listadas abaixo:

a) Avaliação da Qualidade do Produto (AQP): os projetos devem possuir um objetivo quantitativo de qualidade do produto de software definido. Para tanto, é necessário que as necessidades de qualidade sejam bem delineadas e transformadas em objetivos quantitativos de qualidade. Para que os objetivos quantitativos de qualidade sejam atingidos, as estratégias, objetivos e planos do projeto devem ser ajustados para que a direção do projeto de teste esteja em sintonia com os objetivos de qualidade do produto. Esta área de processo é bastante relacionada à área de processo Medição e Análise de Teste, que fornece os mecanismos e infraestrutura de medição para avaliação da qualidade do produto (SOFTEX RECIFE, 2013a, p. 57);
b) Gestão de Defeitos (GDD): muitos defeitos são decorrentes e compartilham a mesma origem. Para determinar a cauza raiz destes defeitos, procedimentos de análise de defeitos devem ser colocados em uso. Após a identificação das causas raiz de defeitos, estas devem ser analisadas, priorizadas e selecionadas de acordo com o custo de sua correção, implicações em não endereçar a causa raiz e consequências na qualidade do produto. Planos de ações contendo ações corretivas são colocados em prática para eliminar estas causas raiz de defeitos, e avaliações de efetividade devem ser conduzidas para garantir que as ações corretivas atenderam a seus objetivos (SOFTEX RECIFE, 2013a, p. 69);
c) Teste não Funcional (TNF): um esforço sistemático deve ser conduzido para endereçar os riscos não-funcionais do produto. Os stakeholders relevantes devem avaliar estes riscos e a estratégia de teste deve ser ajustada para contemplar ações que mitigam estes riscos. A partir dos riscos não-funcionais do produto um projeto de teste não-funcional deve ser aplicado, gerando condições e casos de teste. Estes casos de teste não-funcional devem ser executados e os incidentes identificados devem ser gerenciados até a sua conclusão (SOFTEX RECIFE, 2013a, p. 143).

2.3.2.5 Nível 5 – Automação e otimização

O quinto nível do modelo tem como objetivo estabelecer um processo de melhoria contínua e automação do teste. Dentre as características deste nível, pode-se citar que existe

uma abordagem sistemática para automação da execução do teste, um processo sistemático para seleção e adoção de ferramentas CASE é utilizado, o processo é controlado estatisticamente e está sob melhoria contínua.

As áreas do processo são listadas abaixo:

a) Automação da Execução do Teste (AET): estabelecer e manter uma estratégia para a automação da execução do teste, compreendendo a definição de objetivos, elaboração de um framework e análise do retorno sobre investimento na automação do teste (SOFTEX RECIFE, 2013a, p. 51);
b) Controle Estático do Processo (CEP): organizações de alta maturidade devem possuir mecanismos para entender intimamente o comportamento de seus processos e uma forma de realizar esta análise é através do controle estatístico dos mesmos. Geralmente, apenas uma parte dos processos da organização é colocada sob o CEP, pois ele envolve a aplicação de esforços para planejar, monitorar e controlar estatisticamente o processo, sendo uma atividade que demanda bastante recursos. Sendo assim, nem todos os processos são submetidos ao CEP. Para que ele seja efetuado é necessário identificar objetivos de desempenho de seus processos e selecionar dentro de seu conjunto de processos padrão um subconjunto de processos ou sub-processos que necessitam atender a estes objetivos. Sub-processos são pequenas partes de um processo, por exemplo, as atividades de definição da WBS compõe um sub-processo de planejamento do projeto. Um conjunto de medidas deve ser especificado para avaliar estes processos selecionados e gerar baselines de desempenho que serão usadas para controlar a execução dos projetos que utilizam estes processos. Modelos de desempenho também devem ser gerados para predizer o comportamento dos processos durante o seu uso em novos projetos, produtos e serviços (SOFTEX RECIFE, 2013a, p. 61);
c) Gestão de Ferramentas (GDF): ferramentas de software podem trazer inúmeros benefícios à produtividade em uma organização, reduzindo tempo e custo de execução das atividades. Porém, caso a implantação de ferramentas não seja gerenciada, estas mesmas ferramentas podem se tornar grandes pesadelos. Para evitar que isto ocorra, é necessário realizar um conjunto de ações para verificar a real necessidade de uma ferramenta, realizar uma seleção entre as opções disponíveis, implantar a ferramenta que se sobressair na seleção e avaliar a efetividade da ferramenta (SOFTEX RECIFE, 2013a, p. 73).

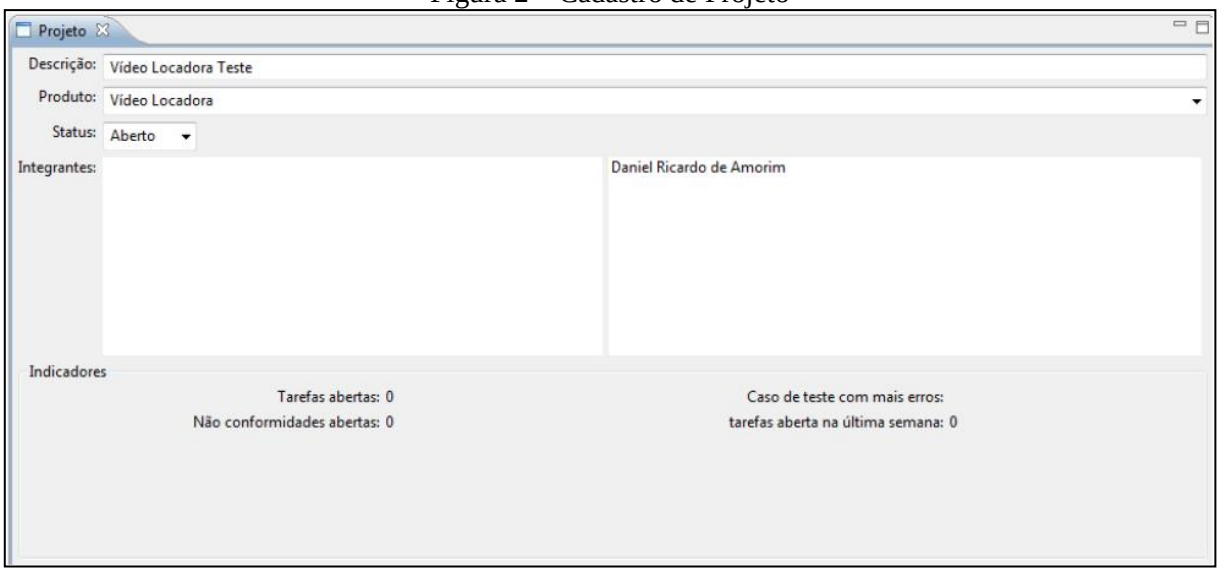
2.4 FERRAMENTA DE APOIO À IMPLEMENTAÇÃO DO PROCESSO MELHORIA DE PROCESSO DE TESTE (MPT)

A ferramenta de apoio à implementação do processo melhoria de processo de teste (MPT) (AMORIM, 2011) tem como principal objetivo o desenvolvimento de uma ferramenta que atenda ao modelo MPT.BR (na sua versão 2.0, existente à época) e como objetivo específico atender aos níveis de maturidade 1 e 2, na qual contempla as áreas de "Gerência de Projetos de Teste (GPT)", "Projeto e Execução de Teste" e "Gerência de Requisitos de Teste" do modelo, sendo uma aplicação desenvolvida em forma de *plug-in* para Eclipse. Os requisitos principais da ferramenta são:

a) permitir revisar documentos de desenvolvimento;
b) permitir elaborar planejamento dos testes;
c) permitir documentar o ambiente de teste necessário;
d) permitir elaborar casos de teste;
e) permitir documentar o resultado dos testes unitários;
f) permitir registrar e acompanhar defeitos;
g) permitir registrar resultados dos testes manuais;
h) permitir controlar quais ferramentas necessárias para o sucesso do projeto através da gerência de configurações do projeto de teste;
i) permitir extrair resultados e medições do projeto através de indicadores;
j) permitir integrar os módulos, sendo que cada nível do processo representa um módulo.

A ferramenta possui alguns cadastros iniciais, como o cadastro de produtos, cargos, usuários e suas respectivas permissões de acesso. O principal ponto da ferramenta fica no cadastro de projeto, pois é neste momento que o usuário grava as informações necessárias referentes ao projeto. Na figura 2 é ilustrado o cadastro de projeto.

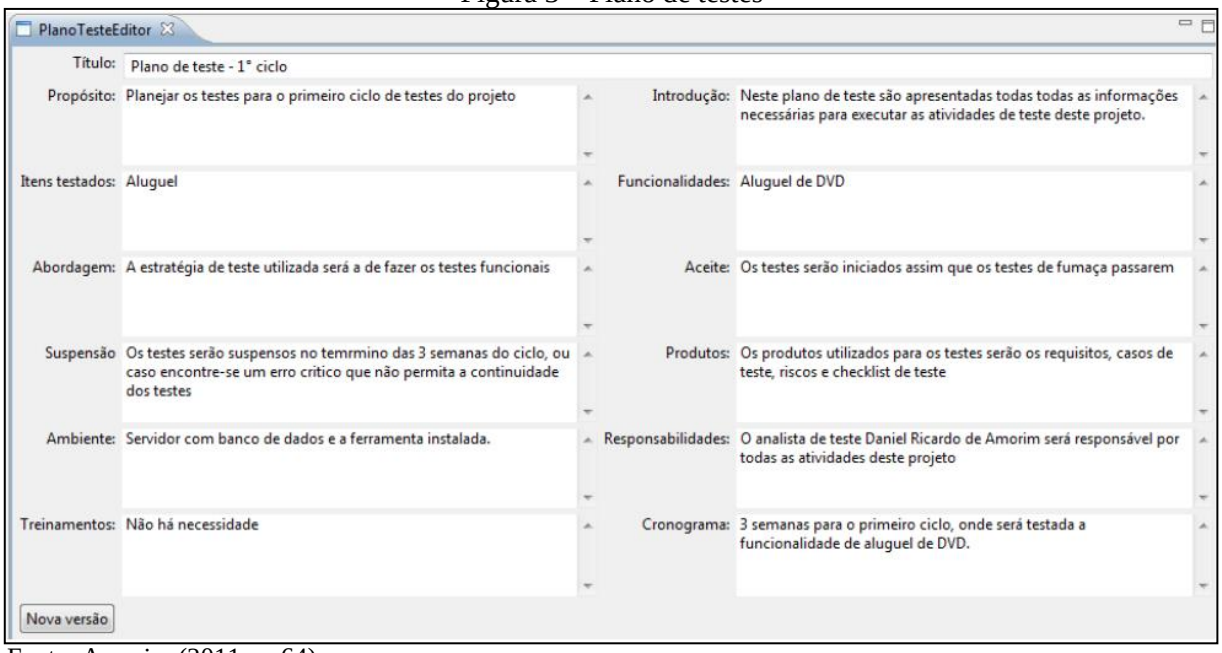
Figura 2 – Cadastro de Projeto



Após o cadastro de projetos, o testador tem a possibilidade de abrir atas de reunião para discussão sobre o projeto em andamento. Pode-se cadastrar os riscos durante o desenvolvimento do projeto que é um ponto crucial, pois há a necessidade de se saber os obstáculos que poderão ocorrer durante o desenvolvimento do projeto.

Outro ponto importante a destacar da ferramenta de Amorim (2011) é o plano de testes, que é o principal instrumento de testes do MPT.BR, é nele que serão anexadas todas as etapas de um processo de teste como por exemplo o cronograma a ser seguido. Segundo Amorim (2011, p. 73), o plano de teste representa o artefato principal do planejamento do projeto de teste. Na figura 3 é ilustrada o plano de testes da ferramenta de Amorim (2011).

Figura 3 – Plano de testes



Fonte: Amorim (2011, p. 64).

Por fim, a ferramenta possibilita extrair indicadores que auxiliam a equipe de testes no desenvolvimento do projeto em questão.

Em relação aos níveis de maturidade 1 e 2 do MPT.BR, a ferramenta não os consegue atender por completo, sendo que algumas práticas atendidas o foram de forma parcial. No quadro 2 são listados os requisitos do modelo atendidos de forma parcial.

Quadro 2 – Requisitos do modelo atendidos de forma parcial

PRÁTICA
Estabelecer estimativas para o tamanho das tarefas e dos produtos de trabalho do projeto de teste utilizando métodos apropriados.
Estimar o esforço e o custo para a execução das tarefas e dos produtos de trabalho com base em dados históricos ou referências técnicas.
Estabelecer e manter o orçamento e o cronograma do projeto de teste, incluindo marcos e/ou pontos de controle.

Fonte: Amorim (2011, p. 73).

Segundo Amorim (2011, p. 27), os resultados obtidos no término do desenvolvimento da ferramenta são satisfatórios e permitem o gerenciamento de projetos de teste.

2.5 BIBLIOTECA PLUG-IN DEVELOPMENT ENVIRONMENT (PDE)

Segundo Eclipse Foundation (2014), o *Plug-in Development Environment* (PDE) é um recurso para Eclipse que oferece ferramentas para criar, desenvolver, testar, depurar, construir e implantar *plug-ins*, recursos, sites de atualização entre outros.

Eclipse Foundation (2014) afirma que o PDE é divido em quatro componentes:

a) PDE *build*: componente para a criação de ferramentas e *scripts* para a automatização da construção de processos;
b) PDE UI: composto por modelos, construtores e editores para facilitar o desenvolvimento de *plug-ins* para o Eclipse;
c) PDE *Application Programming Interface* (API) tools: IDE do Eclipse com ferramentas de integração de criação de processos para manter a API;
d) PDE *incubator*: desenvolvimento de novas ferramentas que não estão prontas para serem adicionadas no Eclipse.

2.6 TRABALHOS CORRELATOS

Existem várias ferramentas não comerciais que apresentam funções similares, mas cada uma com sua particularidade. Dentre elas apresentam-se o XPlanner (2002) e a ferramenta desenvolvida por Meisen (2005).

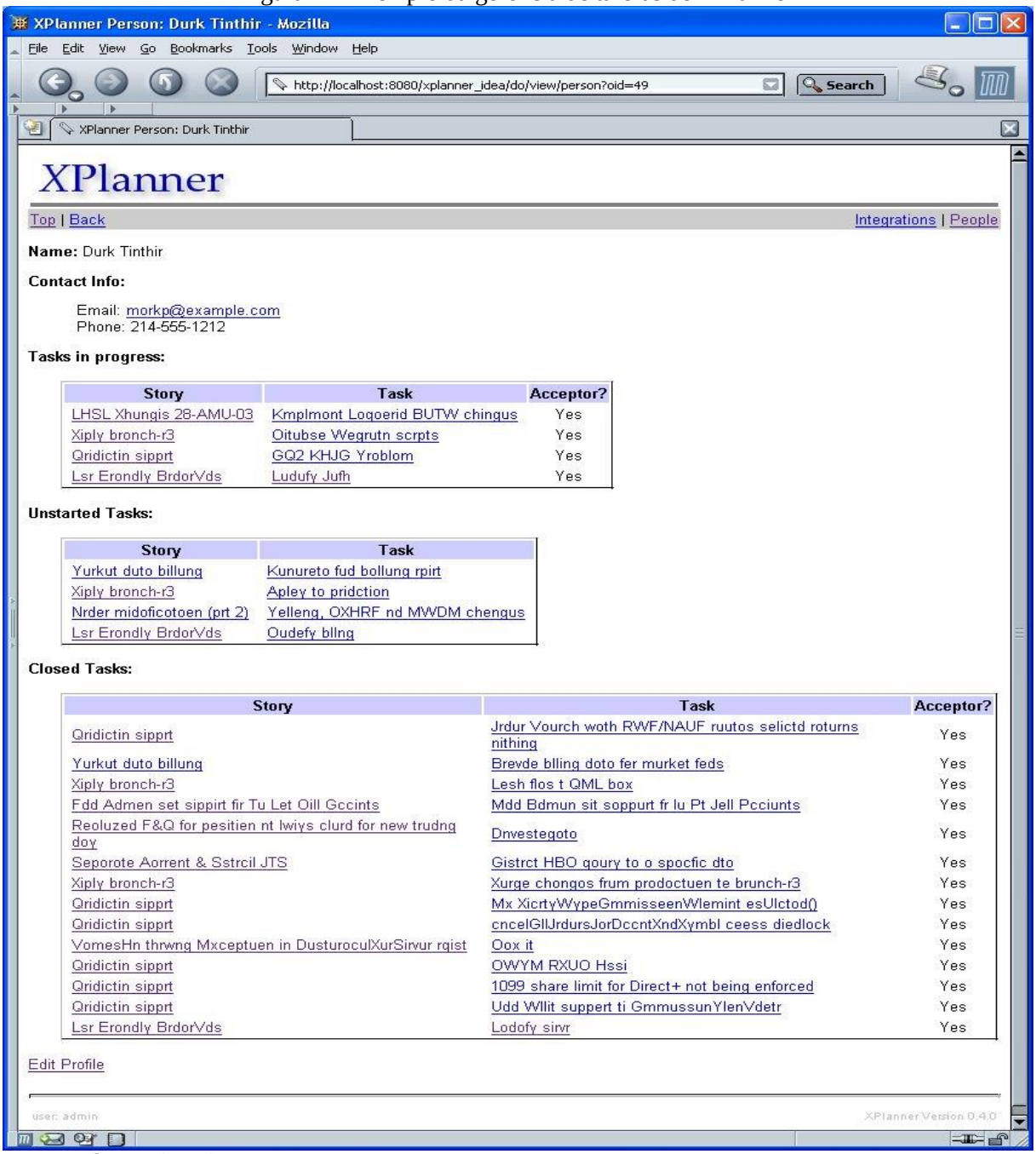
2.6.1 XPlanner

Segundo XPlanner (2002), a ferramenta oferecida gratuitamente pela organização foi criada para atender uma metodologia de desenvolvimento de software chamada *eXtreme Programming* (XP). São características do XPlanner:

a) possuir um modelo simples de planejamento;
b) anotações;
c) suportar a gravação e o monitoramento de projetos, iterações e tarefas;
d) continuar de histórias inacabadas;
e) permitir medições de tempo trabalhado;
f) gerar gráficos de velocidade da iteração;
g) permitir distribuição de tarefas;
h) anexar notas às histórias e tarefas;
i) fornecer uma visão da estimativa exata da iteração;
j) possuir uma página de tarefas, mostrando a história individualmente para desenvolvedores e clientes;
k) exportar projetos para *Extensible Markup Language* (XML), *Portable Document Format* (PDF) e outros;
l) suportar vários idiomas.

Na figura 4 tem-se uma visão da gerência de tarefas no XPlanner.

Figura 4 – Exemplo da gerência de tarefas do XPlanner



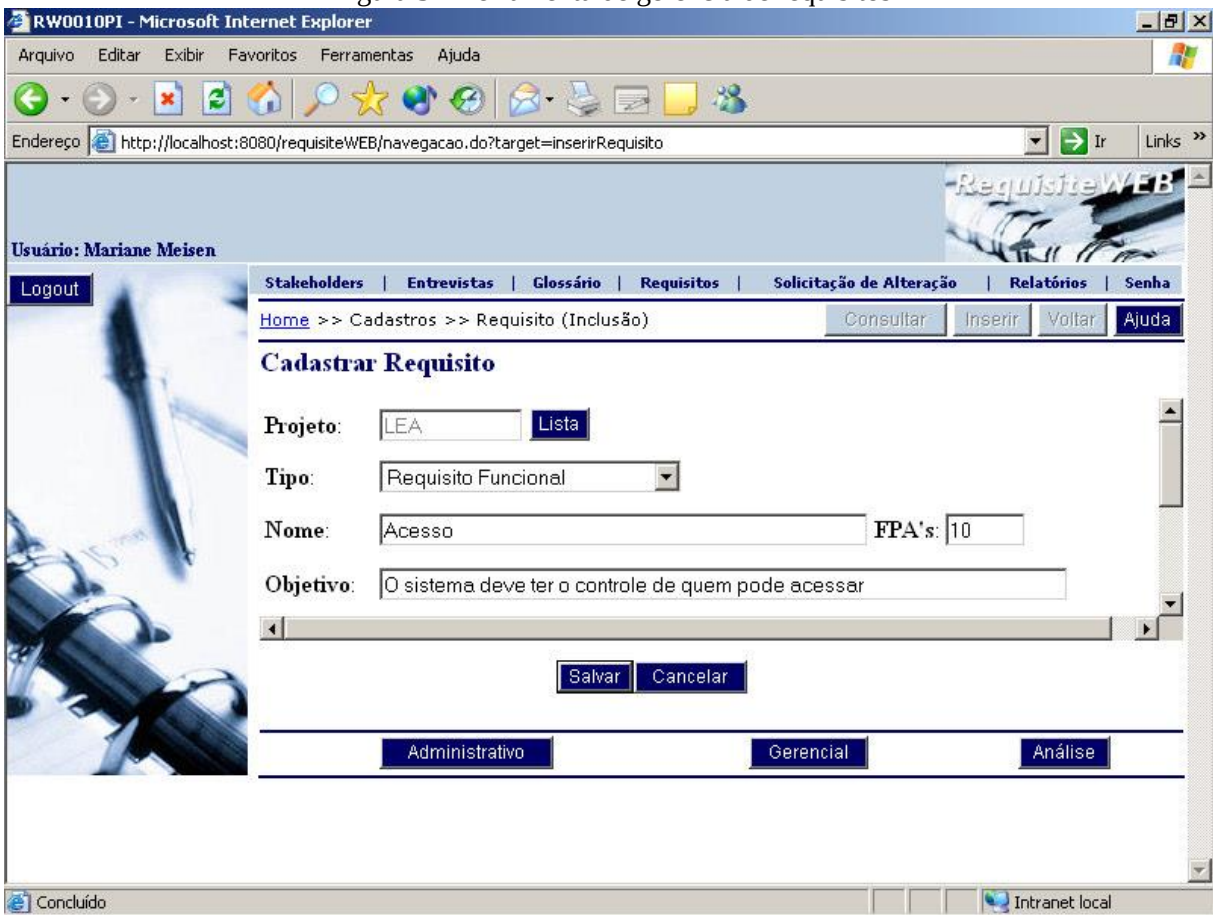
Fonte: XPlanner (2002).

2.6.2 Ferramenta de apoio à gerência de requisitos baseado no Modelo de Maturidade em Capacitação – Integração (CMMI)

Segundo Meisen (2005), a ferramenta tem a função de auxiliar a gerência de requisitos em projetos de software. Foi criada para atender a alguns dos requisitos do modelo CMMI referentes aos níveis 2 e 3, através do desenvolvimento de uma área para a criação de um plano de gerência de requisitos, histórico de mudanças, indicadores de estabilidade e

cadastros básicos para a gerência de requisitos em projetos. Na figura 5 é possível verificar como a ferramenta disponibiliza a funcionalidade de cadastro de requisitos.

Figura 5 – Ferramenta de gerência de requisitos



Fonte: Meisen (2005).

Segundo Meisen (2005), os requisitos da ferramenta proposta são:

a) criar projetos: o sistema deve permitir a criação do projeto;
b) manter usuários: o sistema deve permitir a inclusão/alteração/exclusão de tipos de requisitos que serão utilizados nos projetos;
c) manter tipos de requisitos: o sistema deve permitir a inclusão/alteração/exclusão de tipos de requisitos que serão utilizados nos projetos;
d) manter atributos: o sistema deve permitir a inclusão/alteração/exclusão de atributos que serão associados aos tipos de requisitos;
e) manter vínculos: o sistema deve permitir a inclusão/alteração/exclusão de vínculos entre os tipos de requisitos;
f) manter projetos: o sistema deve permitir a inclusão/alteração/exclusão dos projetos previamente criados;

g) registrar uma política para gerência de requisitos: o sistema deve permitir a criação de um documento que descreva as regras de gerência de requisitos da empresa, para que todos os usuários tenham acesso;
h) manter acessos: o sistema deve permitir a inclusão/alteração/exclusão de acessos de escrita ou leitura aos analistas;
i) manter glossário: o sistema deve permitir a inclusão/alteração/exclusão do glossário dos projetos;
j) manter requisitos: o sistema deve permitir a inclusão/alteração/exclusão dos requisitos.

2.6.3 Comparação entre os trabalhos correlatos e a ferramenta de AMORIM (2011)

Nesta seção é feita uma comparação entre os trabalhos correlatos apresentados anteriormente e a ferramenta de apoio a implementação do processo melhoria de processo de teste (MPT) de AMORIM (2011), baseados nos níveis de maturidade 1, 2 e 3 do MPT.BR a qual comtempla as áreas de "Gerência de Projetos de Teste (GPT)", "Projeto e Execução de Teste (PET)", "Gerência de Requisitos de Teste (GRT)", "Fechamento do Teste (FDT)", "Medição e Análise do Teste (MAT)" e "Teste de Aceitação (TDA)". A mesma pode ser visualizada no quadro 3.

Quadro 3 – Comparação entre os trabalhos correlatos e a ferramenta de AMORIM (2011)

ÁREA	**FERRAMENTA**		
	XPlanner	**Meisen**	**Ferramenta AMORIM (2011)**
Gerência de Projetos de Teste (GPT)	Atende	Atende Parcialmente	Atende parcialmente
Projeto e Execução de Teste (PET)	Não atende	Atende parcialmente	Atende parcialmente
Gerência de Requisitos de Teste (GRT)	Atende	Atende	Atende
Fechamento do Teste (FDT)	Atende	Atende parcialmente	Atende parcialmente
Medição e Análise do Teste (MAT)	Atende	Atende	Atende parcialmente
Teste de Aceitação (TDA)	Atende	Atende parcialmente	Não atende

3 DESENVOLVIMENTO

Nesta seção são apresentados os requisitos principais da ferramenta, as especificações utilizadas para a implementação do trabalho, a implementação do *plug-in* e, por fim, os resultados e discussões obtidos.

3.1 REQUISITOS PRINCIPAIS DA FERRAMENTA

Os requisitos especificados para a ferramenta desenvolvida são baseados no nível 3 do modelo MPT.BR.

A ferramenta deverá permitir:

a) registrar os artefatos de teste para uso em outros projetos (Requisito Funcional – RF);
b) gerar relatório com informações sobre a execução do teste (RF);
c) registrar os produtos do software que serão avaliados (RF);
d) registrar os critérios de aceitação para cada tipo de produto (RF);
e) registrar um plano de aceitação (RF);
f) revisar se o ambiente de testes está configurado conforme plano de testes (RF);
g) gerar relatório com os incidentes encontrados na revisão de configuração do plano de teste (RF);
h) utilizar o recurso PDE para sua construção (Requisito Não Funcional – RNF);
i) utilizar a linguagem Java para sua construção (RNF);
j) utilizar o banco de dados MySQL para persistir os dados (RNF);
k) utilizar a biblioteca `iText` para a geração dos relatórios em formato PDF (RNF).

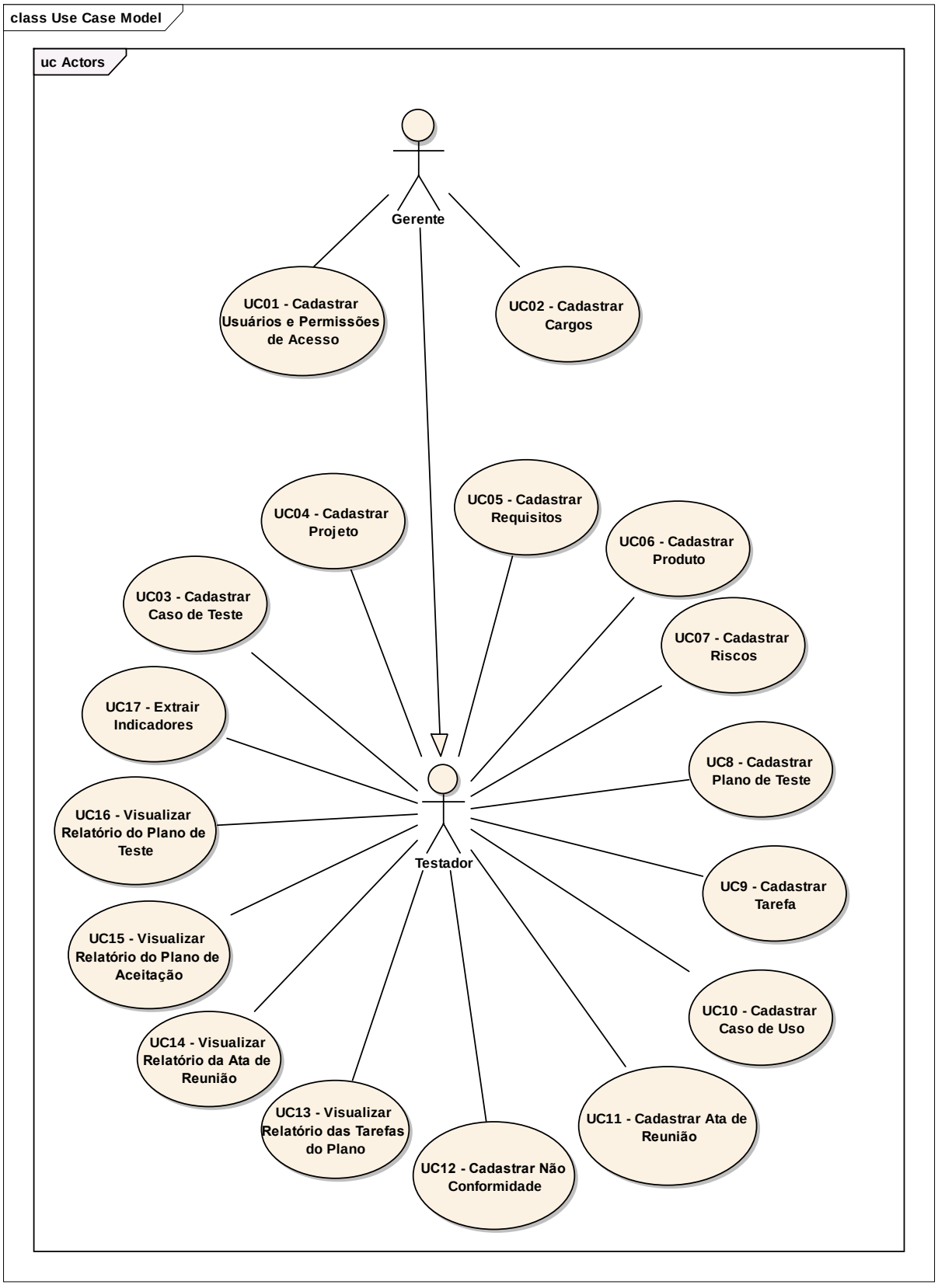
3.2 ESPECIFICAÇÃO

Nesta seção serão abordadas as especificações para a criação da ferramenta, diagrama de classes, diagrama de casos de uso e diagrama de relacionamento de tabelas.

3.2.1 Diagrama de casos de uso

Na figura 6 é visualizado o diagrama de casos de uso utilizado para a criação da ferramenta.

Figura 6 – Diagrama de Casos de Uso



O diagrama de casos de uso apresenta dois atores: o gerente, que irá definir os papéis de cada usuário envolvido no projeto de testes, juntamente com as permissões que cada um

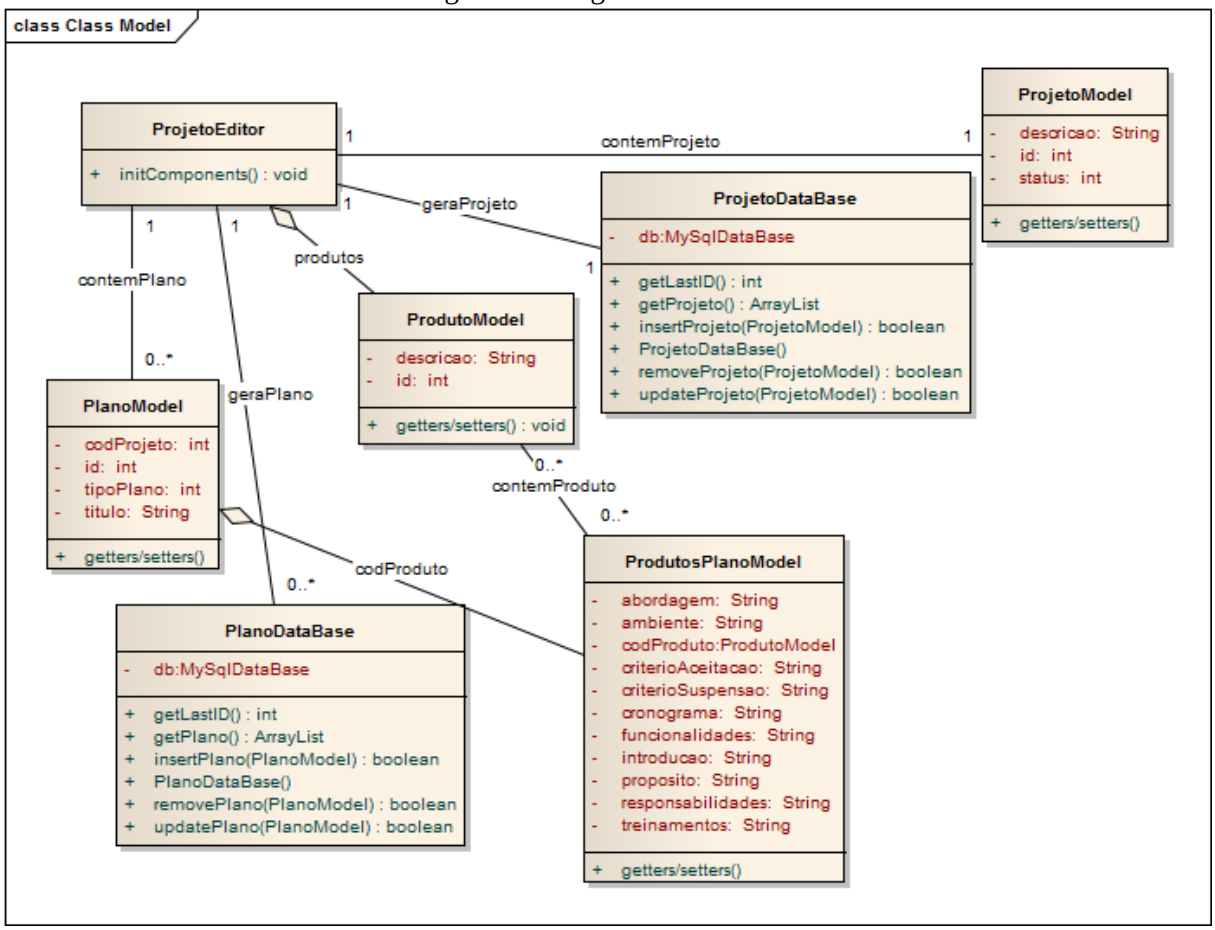
terá; e por fim o testador, que irá executar as tarefas referentes ao modelo MPT.BR, mediante permissões individuais para cada tarefa que deverá ser efetuada.

O detalhamento completo de cada caso de uso apresentado na figura 5, está descrito no apêndice A.

3.2.2 Diagrama de Classes

Na figura 7 é ilustrado o diagrama de classes. São descritas apenas as classes que compõem o desenvolvimento do plano de testes ou de aceitação pertencente a determinado projeto.

Figura 7 – Diagrama de Classes



A classe `ProjetoEditor` é a responsável por gerar a interface gráfica. Ela possui um único método `initComponents` que carrega todos os componentes da tela.

A classe `ProjetoModel` contém todas as informações que compõem um projeto. Nela é possível saber se um projeto ainda está em execução ou não.

A classe `ProjetoDataBase` é a responsável pela comunição com o banco de dados. Ela captura as informações da tabela correspondente e, cria um objeto `ProjetoModel` que é acessado na classe `ProjetoEditor`.

A classe `ProdutoModel` contém as informações gerais do produto e é utilizada em praticamente toda a ferramenta.

A classe `PlanoModel` tem vínculo direto com a classe `ProjetoEditor`, de forma que um projeto pode conter diversos planos. Ela contém informações gerais, tais como a de que se o plano é de testes ou de aceitação.

A classe `PlanoDataBase` é a responsável pela comunicação com o banco de dados. Busca informações da tabela correspondente e cria um objeto `PlanoModel` que é acessado na classe `ProjetoEditor`.

A classe `ProdutosPlanoModel` traz as informações específicas do plano e tem vínculo direto com a classe `PlanoModel`, pois um plano pode conter diversos produtos com as suas respectivas informações.

3.3 IMPLEMENTAÇÃO

A seguir são apresentadas todas as técnicas e ferramentas utilizadas. Também é explicado o aspecto operacional da implementação através de um estudo de caso.

3.3.1 Técnicas e ferramentas utilizadas

Nesta seção são descritas as técnicas utilizadas, apresentando a arquitetura da plataforma Eclipse para criação de *plug-ins*, o ambiente PDE, e o MySQL utilizado como banco de dados.

3.3.1.1 Arquitetura da plataforma Eclipse

Segundo Pedrosa et al. (2004), a plataforma Eclipse é estruturada em torno do conceito de pontos de extensão. Pontos de extensão são lugares no sistema bem definidos onde outras ferramentas (chamadas de *plug-ins*) podem contribuir com funcionalidades.

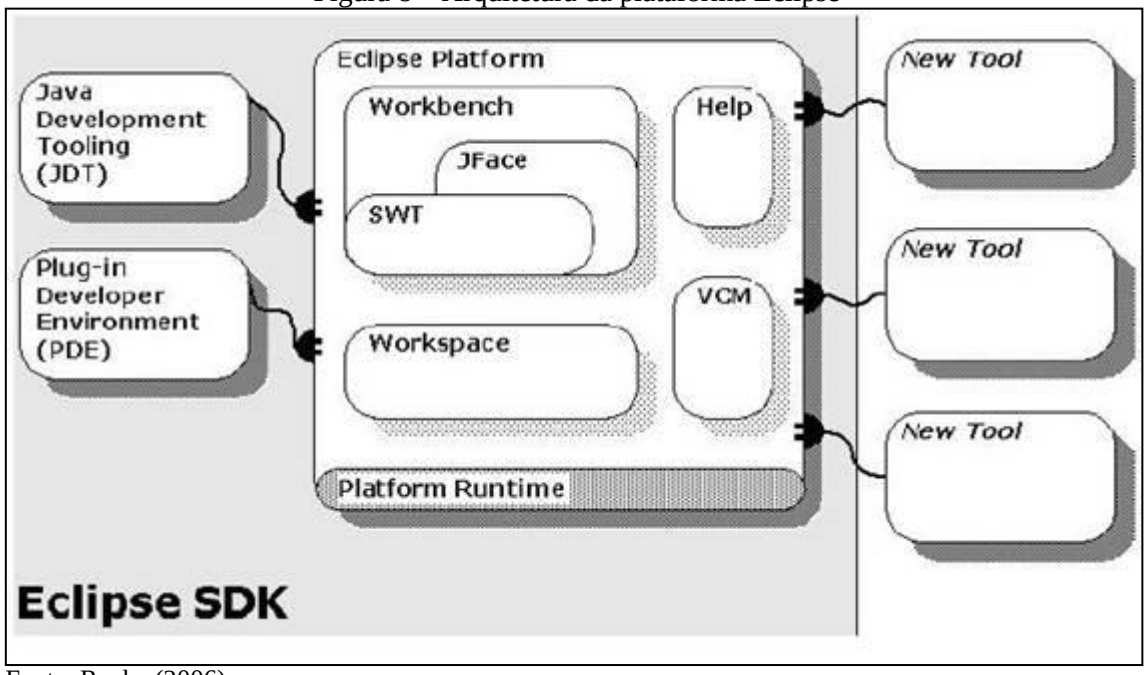
Cada sistema principal na plataforma é estruturado como um conjunto de *plug-ins* que implementam alguma função chave e definem pontos de extensão. *Plug-ins* podem definir seus próprios pontos de extensão ou simplesmente adicionar extensões aos pontos de extensão de outros *plug-ins*.

Segundo Eclipse Foundation (2010), os subsistemas da plataforma adicionam inúmeras características e fornecem APIs para estender sua funcionalidade. Alguns destes componentes fornecem as bibliotecas de classe adicionais que não se relacionam diretamente a um ponto de extensão, mas podem ser usados para implementar extensões. Por exemplo, a *workbench*

fornece a *JFace UI* e o conjunto de ferramentas *widget* (símbolo gráfico de uma interface que possibilita a interação entre o usuário e o computador) do *Standard Widget Tools* (SWT).

Segundo Rocha (2006), o *Software Development Kit* (SDK) do Eclipse, inclui a plataforma básica mais duas ferramentas principais que são úteis para o desenvolvimento de *plug-ins*. A ferramenta de desenvolvimento *Java Development Tools* (JDT) implementa um ambiente de desenvolvimento Java rico em recursos. O ambiente de desenvolvimento de *plug-ins* (PDE) adiciona as ferramentas especializadas que possibilitam o desenvolvimento de *plug-ins* e extensões. Na figura 8 é ilustrada a arquitetura da plataforma Eclipse

Figura 8 – Arquitetura da plataforma Eclipse



Fonte: Rocha (2006).

3.3.1.2 Plug-in Development Environment

Para a ferramenta desenvolvida foi utilizado o PDE, que permite desenvolver *plug-ins* para o IDE Eclipse. A principal motivação para isso foi tornar o IDE uma ferramenta única, ou seja, permitir que o usuário não necessite de ferramentas adicionais para executar os procedimentos de teste do modelo MPT.BR. Além disso, como a linguagem de programação utilizada para desenvolver toda a lógica de operação foi Java, escolheu-se desenvolver a ferramenta no estilo arquitetural de um *plug-in*.

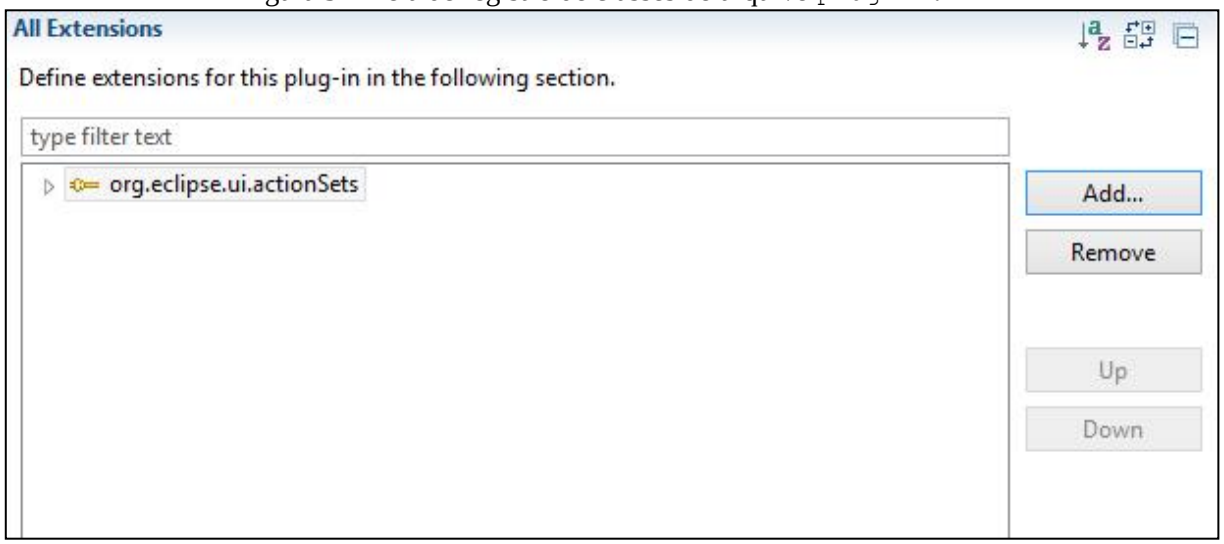
Esta arquitetura agrega uma série de particularidades, as quais serão apresentadas ao longo desta seção.

Ao criar um *plug-in*, é gerado um arquivo `plug-in.xml`. Todas as classes criadas e pertencentes ao *plug-in* devem ser registradas neste arquivo, para que o Eclipse consiga

reconhecê-la como membro pertencente do projeto. Outro ponto importante é, toda classe criada deve possuir um atributo ID, o qual serve como uma espécie de identificação no arquivo `plug-in.xml`.

Na figura 9 apresenta-se a tela de registro de classes do arquivo `plug-in.xml`.

Figura 9 – Tela de registro de classes do arquivo `plug-in.xml`



Ao criar o *plug-in*, a primeira classe gerada é a `Activator`, responsável por iniciar e finalizar o *plug-in* no Eclipse, através dos métodos `start` e `stop`. Esses métodos permitem que o *plug-in* possa salvar e restaurar quaisquer informações dentro das sessões do Eclipse.

É através do método `start` que a ferramenta consegue identificar se o usuário e senha estão corretos para assim executar o *plug-in*.

Na figura 10 apresenta-se a classe `Activator`.

Figura 10 – Classe `Activator`

```
public class Activator extends AbstractUIPlugin {

    // The plug-in ID
    public static final String PLUGIN_ID = "MPT"; //$NON-NLS-1$

    // The shared instance
    private static Activator plugin;

    /**
     * The constructor
     */
    public Activator() {
    }

    /*
     * (non-Javadoc)
     * @see org.eclipse.ui.plugin.AbstractUIPlugin#start(org.osgi.framework.BundleContext)
     */
    public void start(BundleContext context) throws Exception {
        super.start(context);
        plugin = this;
        UsuarioLogado.verificaLogin
        (Activator.getDefault().getPreferenceStore().getString(PreferenceConstants.P_USUARIO),
        Activator.getDefault().getPreferenceStore().getString(PreferenceConstants.P_SENHA));
    }
    /*
     * (non-Javadoc)
     * @see org.eclipse.ui.plugin.AbstractUIPlugin#stop(org.osgi.framework.BundleContext)
     */
    public void stop(BundleContext context) throws Exception {
        plugin = null;
        super.stop(context);
    }
```

3.3.1.3 Banco de dados

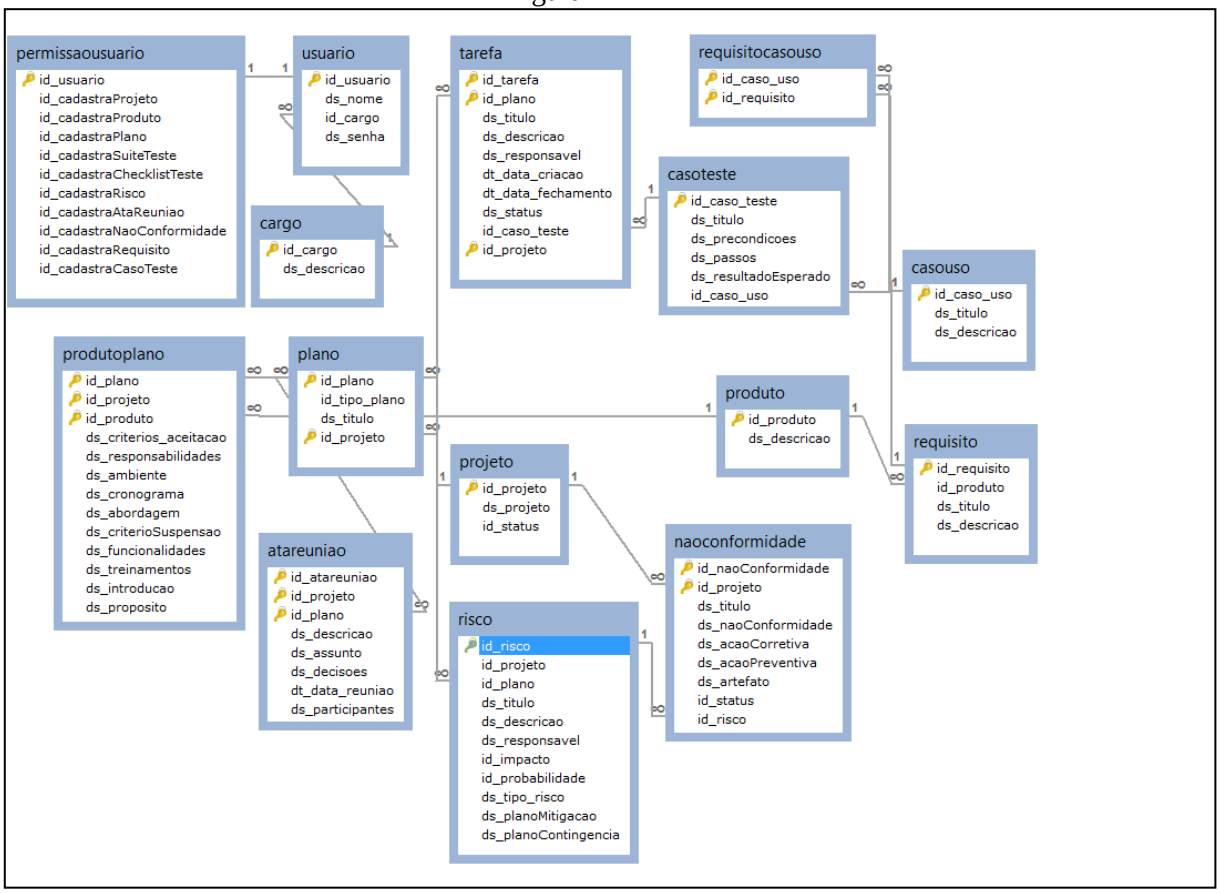
O banco de dados utilizado foi o MySQL, pelo fato de ele ser gratuito e de código fonte aberto, além da facilidade de manipulação. Outro fator importante no uso do MySQL é que ele está disponível para diversos sistemas operacionais diferentes e é amplamente utilizado em sites web, o que torna o seu uso bem popular.

Alecrim (2008) afirma que o MySQL possui alta compatibilidade com a linguagem Java, e possui baixa exigência de processamento.

Hutters (2010) afirma também, que o MySQL possui poder de alta escalabilidade e uma relação custo-benefício atrativa, visto que as aplicações podem manter o mesmo ritmo com o aumento de volume de dados.

O Modelo Entidade-Relacionamento (MER) criado para a ferramenta está descrito na figura 11.

Figura 11 – MER



Ao todo foram criadas 15 tabelas, as quais são descritas a seguir. A tabela `cargo` possui a função de armazenar todas as funções que serão realizadas no ambiente de teste. A tabela `usuario` guarda todos os usuários que irão acessar o ambiente de teste do modelo MPT.BR. A tabela `permissaousuario` vai servir para gravar as permissões que cada usuário vai ter no ambiente de testes, evitando que um usuário execute instruções de outro responsável.

As demais tabelas referem-se a estrutura do modelo de testes MPT.BR nos níveis 1,2 e 3 de maturidade. A tabela `projeto` guarda as informações iniciais referentes ao projeto que irá se dar início. Em seguida, tem-se a tabela `plano` que tem ligação direta com a tabela `projeto`. A tabela `plano` guarda todos os planos de teste e aceitação referentes ao projeto que está sendo executado. A tabela `produtoplano` guarda o produto que está sendo testado no plano aberto.

Cada plano pode ter diversas reuniões. Para isso foi criada a tabela `atareuniao`, que guarda todas as informções das reuniões realizadas para o plano. Cada plano aberto gera uma série de tarefas que serão executadas por algum responsável. Para isso foi criada a tabela `tarefa` que funciona como uma espécie de checklist de serviços realizados para o plano aberto. À cada tarefa aberta é anexado um caso de teste. Para tal, foi criada a tabela

`casoteste`, que grava um caso de teste que deverá ser executado no teste do software, e que descreve o resultado esperado após a sua execução. Para cada caso de teste aberto, é anexado um caso de uso, que é gravado na tebela `casouso`. Por sua vez, esse relaciona-se a um ou mais requisitos. Para esses foram criadas duas tabelas: a tabela `requisito` que guarda todos os requisitos usados em todos os planos de vários projetos; e a tabela `requisitocasouso`, que guarda os requisitos para o caso de uso específico ligado ao caso de teste. A tabela `produto` tem a função de guardar todos os produtos usados nos projetos criados.

Por fim, tem-se as tabelas `risco` e `naoconformidade`, sendo que a primeira armazena todos os riscos que poderão ocorrer durante o projeto, como por exemplo a saída de um testador. A segunda tem por função gravar os problemas encontrados durante o projeto. Esta última é de fundamental importância, para que o gerente de testes saiba o que precisa ser melhorado no projeto antes de entregar o produto final ao cliente.

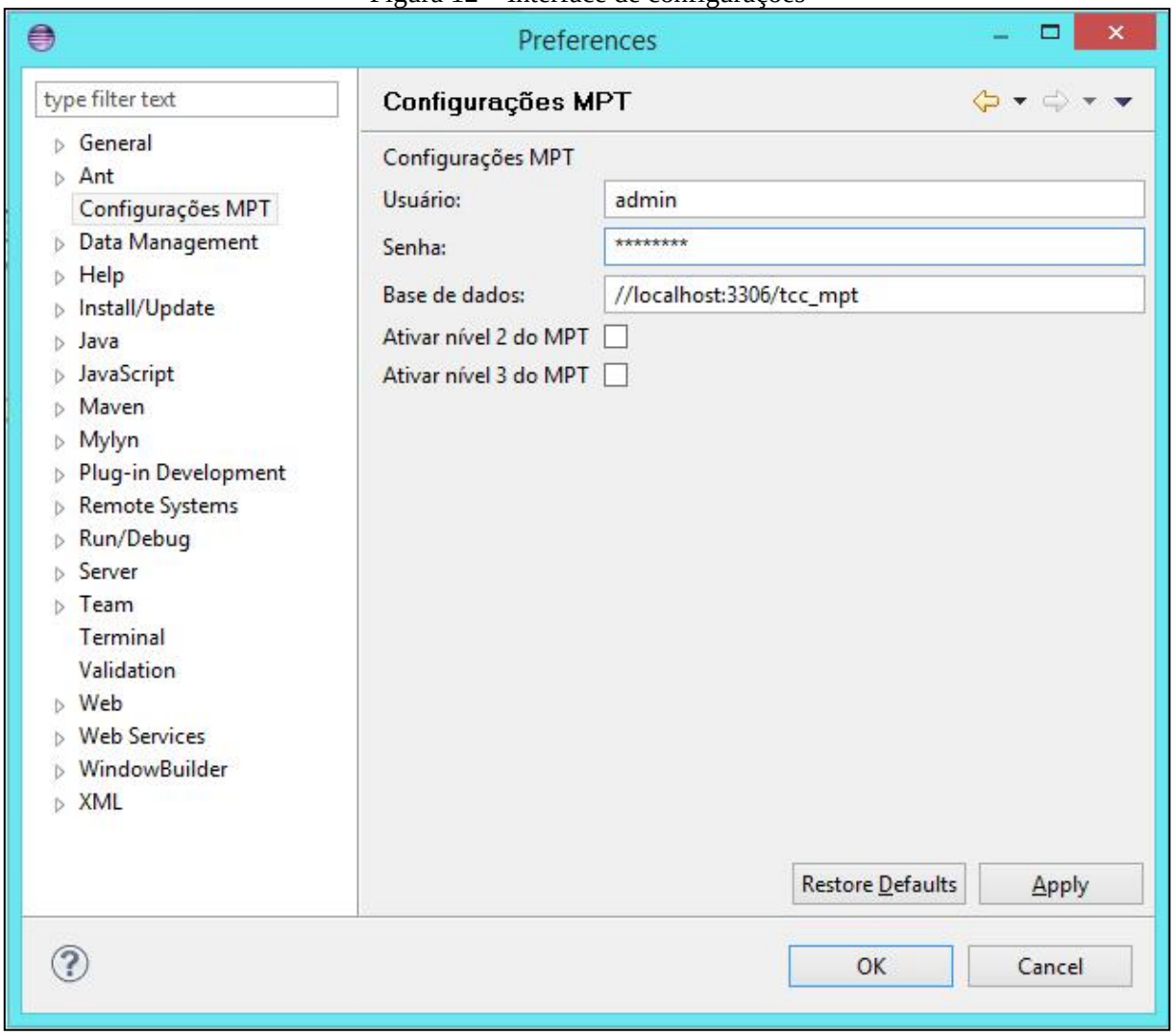
3.3.2 Operacionalidade da implementação

Nesta seção é apresentado um estudo de caso demonstrando as funcionalidades da ferramenta, através de um projeto de teste de um sistema fictício para vídeo locadoras.

3.3.2.1 Configurar MPT

Para que um ou mais usuários possa acessar a base de dados do MPT.BR no MySQL é preciso efetuar uma configuração inicial no ambiente do *plug-in* carregado. Para acessar a interface de configurações deve-se acessar o menu window, preferences e logo em seguida clicar em configurações MPT. Ao fazer isso a interface de configurações (figura 12) é aberta.

Figura 12 – Interface de configurações

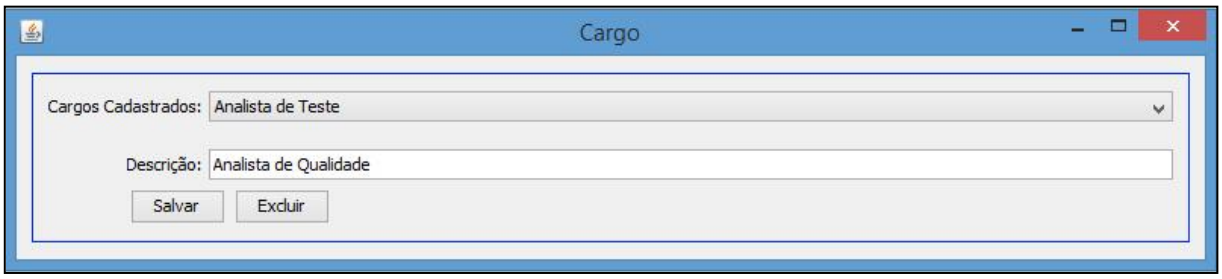


Nesta interface tem-se o campo `usuário`, onde informa-se o usuário do banco de dados, em seguida no campo `senha`, informa-se a senha do usuário da base de dados e por fim no campo `base de dados` se informa a localização da base de dados, neste exemplo a base de dados se encontrava localmente, em seguida seleciona-se quais níveis do MPT se deseja ativar, logo após é clicado no botão ok para criar a conexão com o banco de dados.

3.3.2.2 Cadastrar cargos

Para atender ao caso de uso `UC02 – Cadastrar Cargos`, foi desenvolvida a interface Cargo (figura 13), que tem por finalidade cadastrar os cargos de todos os envolvidos no projeto.

Figura 13 – Interface Cargo

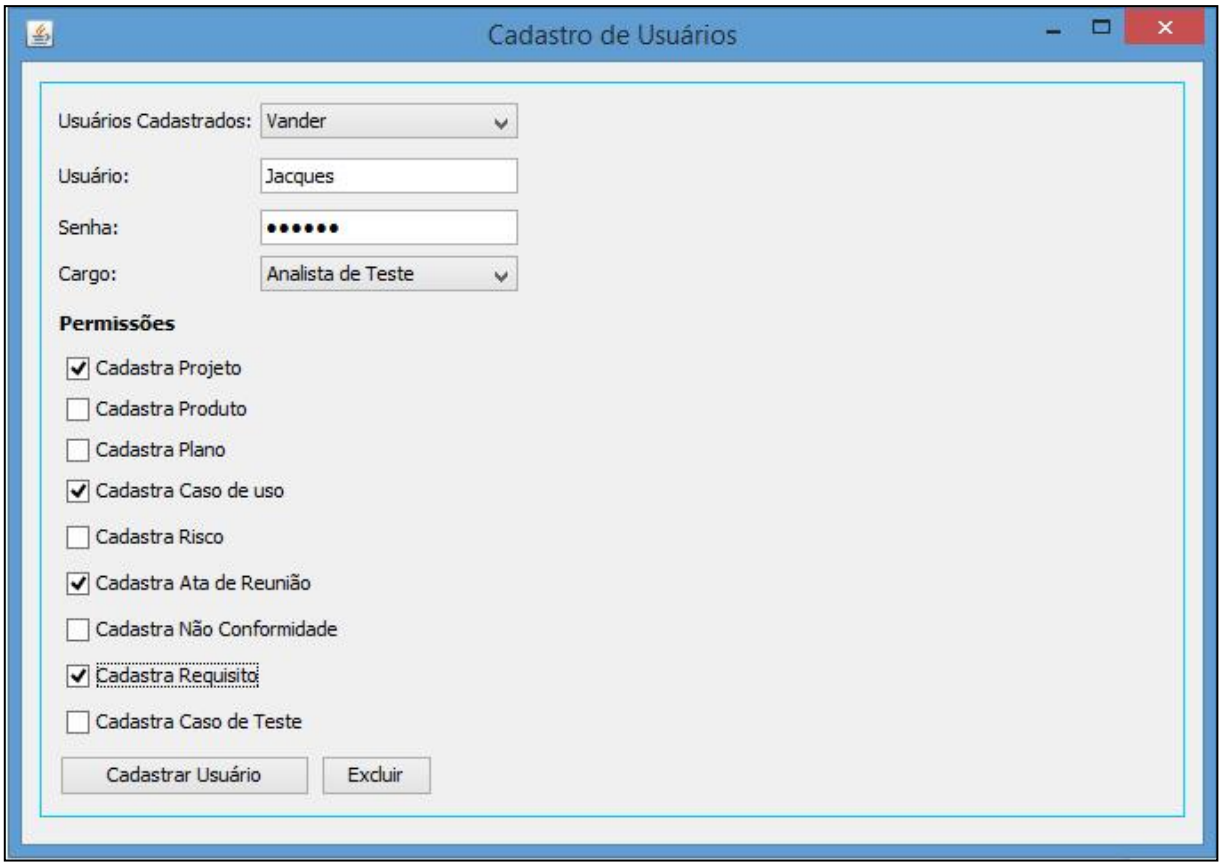


Esta interface, somente terá ações disponíveis para o gerente do projeto, que poderá adicionar ou alterar qualquer cargo. Para cadastrar um novo cargo, basta digitar o nome do cargo no campo `descrição` e em seguida clicar no botão salvar. Caso o campo `descrição` fique em branco, o sistema emite um alerta e não prossegue. Para atualizar o cargo, basta selecionar um daqueles mostrados na opção cargos cadastrados e em seguida digitar as modificações no campo `descrição`. Logo após, clicar em salvar. Para a exclusão segue o mesmo procedimento anterior, só que em vez de clicar sobre o botão salvar, deve-se clicar no botão excluir.

3.3.2.3 Cadastro de usuários e permissões de acesso

Para atender ao casos de uso `UC01 – Cadastrar Usuários e Permissões de Acesso`, foi desenvolvida a interface cadastro de usuários (figura 14), que têm por finalidade cadastrar usuário e senha e, em seguida, as permissões que ele terá.

Figura 14 – Interface Cadastro de Usuários



O campo `usuários` cadastrados lista todos os usuários já inseridos e serve para que o gerente do projeto possa efetuar qualquer alteração ou exclusão de um usuário já cadastrado. No campo `usuário` é informado o nome de usuário que será usado na ferramenta. No campo `senha` deve ser inserida a identificação do respectivo usuário no sistema. O campo `cargo` lista todos os cargos cadastrados na interface cargo. Em seguida tem-se as permissões que o usuário terá. É possível cadastrar uma ou mais permissões de acesso. Somente o gerente do projeto poderá efetuar modificações nesta interface.

3.3.2.4 Cadastro de produtos

Para atender ao caso de uso de `UC06 – Cadastrar Produto` foi desenvolvida a interface produto (figura 15), que tem por finalidade cadastrar todos os produtos que serão anexados nos projetos futuros.

Figura 15 – Interface Produto

O campo `produtos` lista todos os produtos cadastrados até o momento. É possível através dele selecionar qual produto o usuário deseja atualizar ou excluir. O campo `descrição` é utilizado para efetuar o cadastro de um novo produto. Caso esse campo fique em branco e o usuário clique em salvar, o sistema emite mensagem de erro e não efetua o cadastro. O testador tem acesso a esta interface mediante permissão de acesso efetuada pelo gerente do projeto.

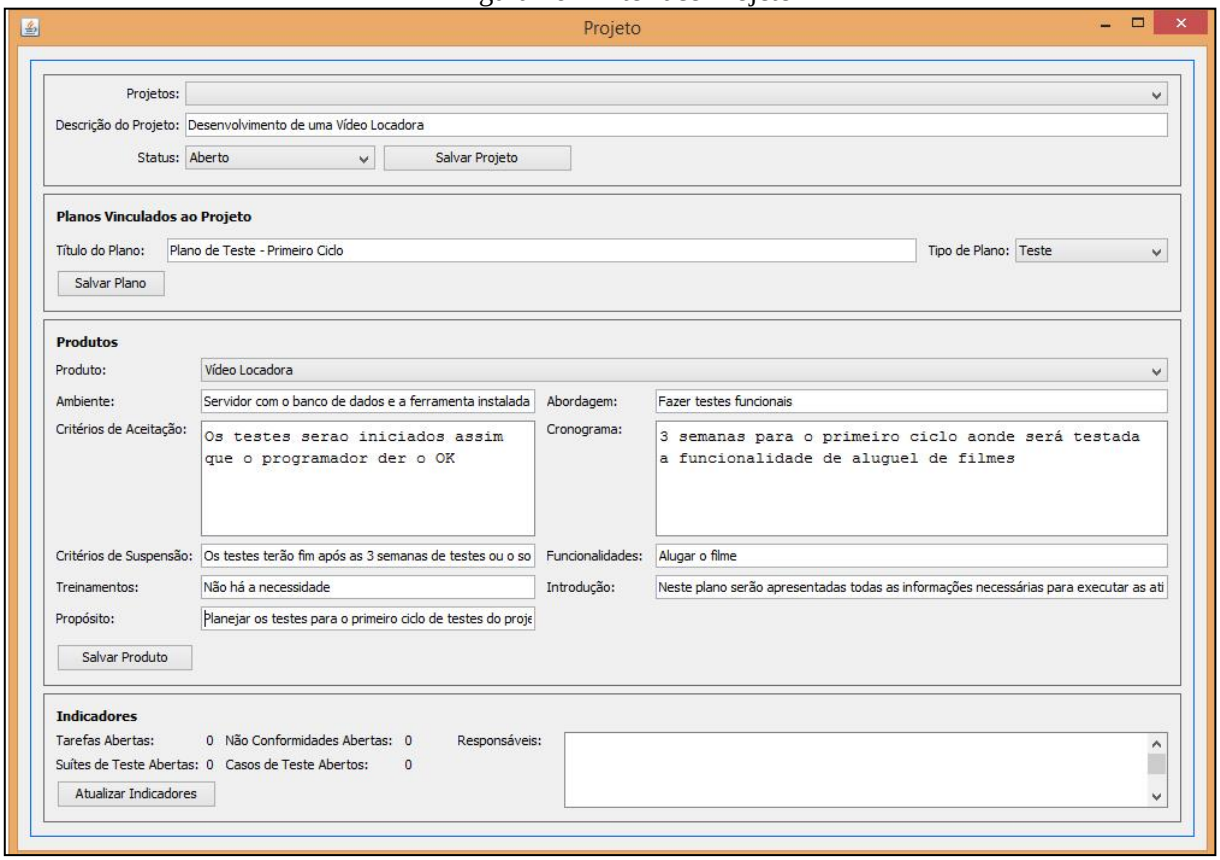
3.3.2.5 Cadastro de projetos e plano

Segundo Amorim (2011), o artefato plano de teste propõe-se a auxiliar o atendimento de algumas práticas do MPT.BR, sendo elas:

a) definir o escopo do trabalho para o projeto;
b) estabelecer estimativas para o tamanho das tarefas e dos produtos de trabalho do projeto de teste utilizando métodos apropriados;
c) estimar o esforço e o custo para a execução das tarefas e dos produtos de trabalho com base em dados históricos ou referências técnicas;
d) estabelecer e manter o orçamento e cronograma do projeto, incluindo marcos e/ou pontos de controle;
e) planejar os recursos humanos para o projeto considerando o perfil e a proficiência necessária para executá-lo;
f) planejar as tarefas, os recursos não humanos e o ambiente de trabalho necessário para executar o projeto de teste;
g) identificar e planejar os artefatos e dados relevantes do projeto de teste quanto à forma de coleta, armazenamento e distribuição;
h) estabelecer os planos para a execução do projeto de teste e reunir no plano.

Para atender ao caso de uso `UC04 – Cadastrar Projeto` e `UC8 – Cadastrar Plano de Teste`, foi desenvolvida a interface projeto (figura 16), sendo esta a principal tela da ferramenta desenvolvida.

Figura 16 – Interface Projeto



A interface projeto é dividida em 4 blocos:

a) projeto: contém as informações iniciais para abertura do projeto, como a `descrição do projeto`, `status`, aberto ou fechado e um botão salvar para gravar as informações. Para definir se um projeto está aberto ou fechado, deve haver permissão específica liberada pelo gerente do projeto;

b) planos vinculados ao projeto: neste bloco é definido se o plano será de testes ou de aceitação e o título do plano, por fim um botão salvar para gravar as alterações. Também é possível adicionar planos futuros, ou seja, o testador seleciona a qual projeto deseja anexar o plano, através do campo `projetos`;

c) produtos: este bloco é um complemento do plano aberto, pois o usuário pode adicionar um ou mais produtos ao plano aberto. Os campos do plano são:

   - `produto`: representa o artefato externo que influencia no planejamento dos teste;
   - `ambiente`: representa o ambiente necessário para a realização dos testes, contado desde o ambiente físico, estações de trabalho, ferramentas e softwares;
   - `abordagem`: representa a estratégia de testes utilizada no plano de teste ou de aceitação e que será utilizada no teste do produto durante o projeto;

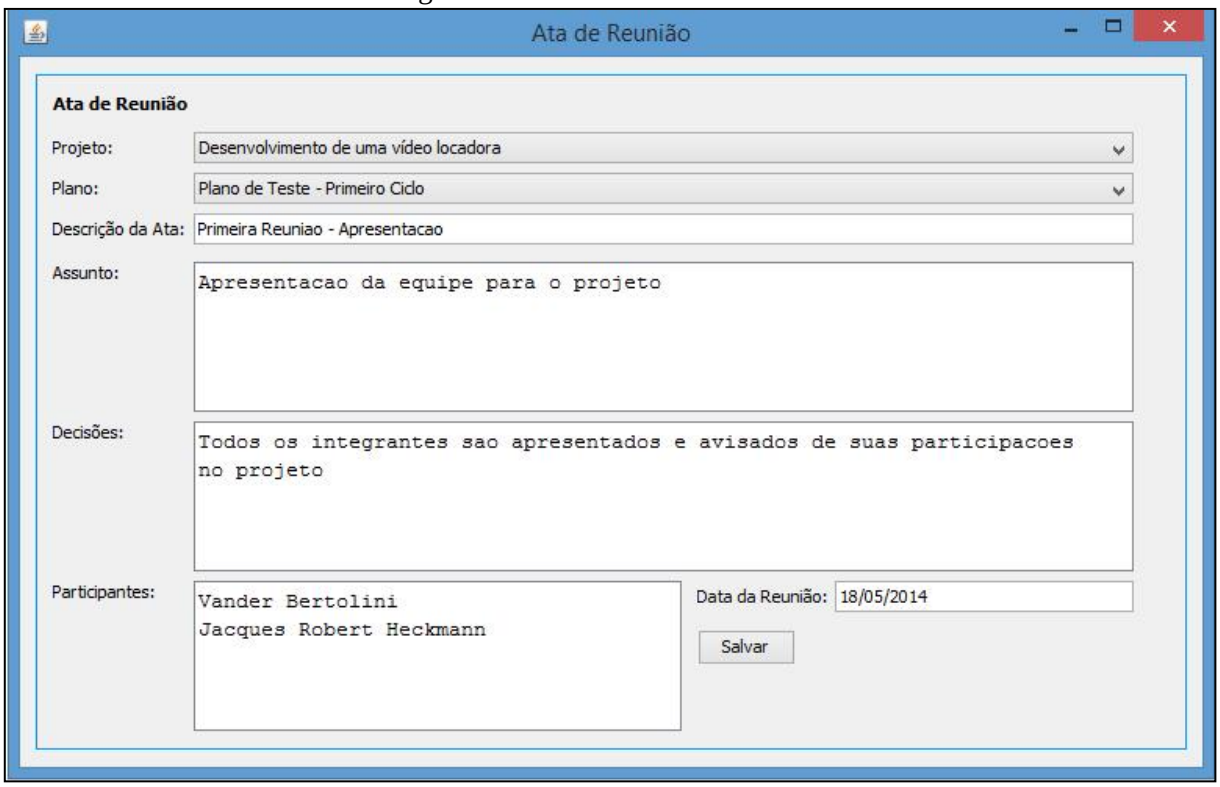
- `critérios` de aceitação: representa as condições na qual é aceita a liberação de um produto;
- `cronograma`: representa os tempos adquiridos para realização do plano;
- `critérios de suspensão`: representa os casos em que serão suspensas as atividades do plano;
- `funcionalidades`: representa as funções englobadas no plano;
- `treinamentos`: se há a necessidade de realizar algum treinamento para a equipe do projeto;
- `introdução`: apresenta uma breve descrição do que abordará o plano aberto;
- `propósito`: o porque do plano existir;

d) indicadores: é através do projeto, que são agendadas tarefas, não conformidades e casos de teste, sendo estes explicados mais detalhadamente adiante. Neste ponto o objetivo é que o testador ou gerente saibam o que já ocorreu com o projeto que está em execução.

3.3.2.6 Cadastrar ata de reunião

Para atender ao `UC11 – Cadastrar Ata de Reunião` foi desenvolvida a interface ata de reunião, visualizada na figura 17.

Figura 17 – Interface Ata de Reunião



A ata de reunião está diretamente ligada ao plano aberto para o projeto. Sendo assim, é possível gerar quantas reuniões forem necessários para o projeto e para o plano. O formulário de cadastro é composto pelos seguintes campos:

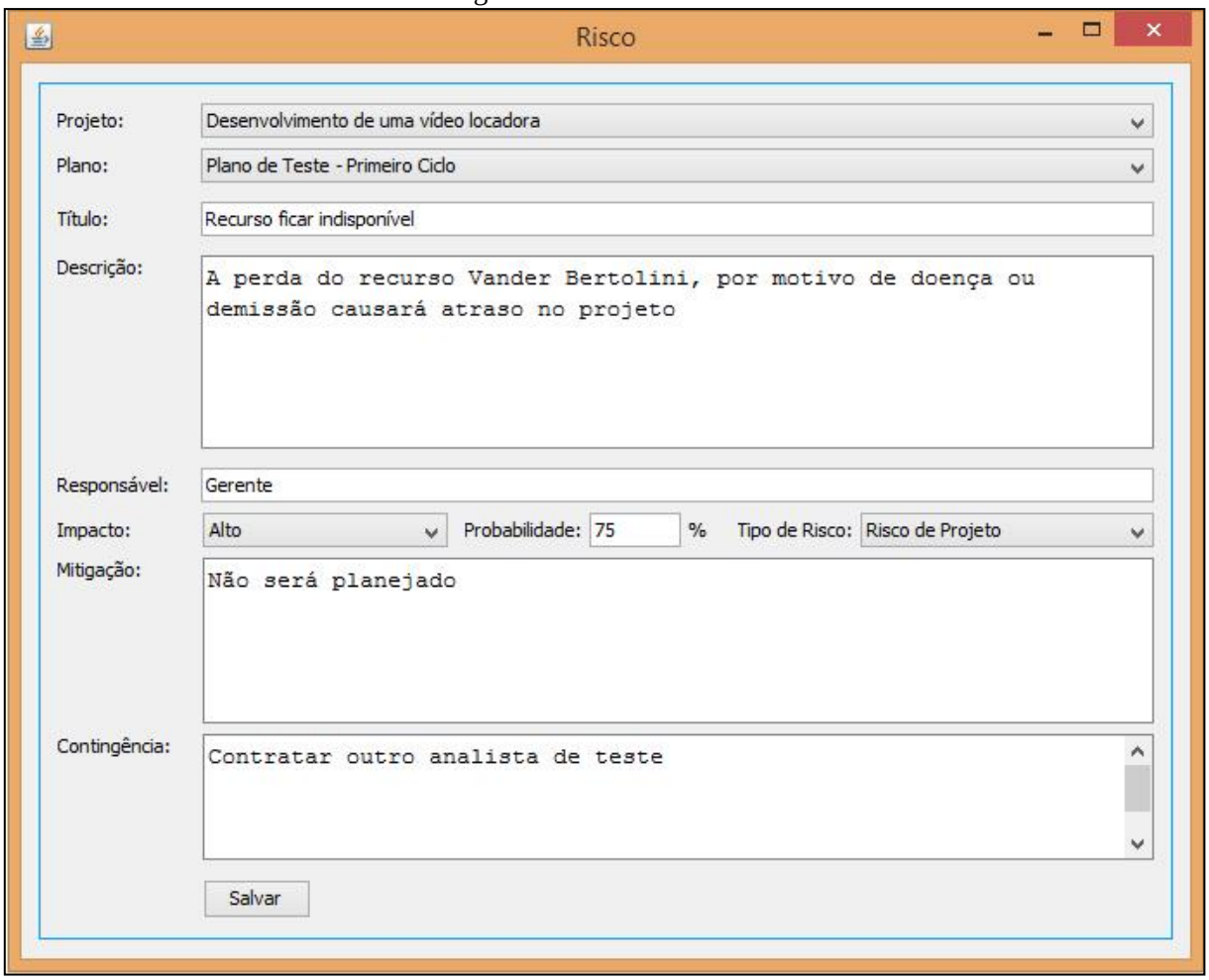
a) `projeto`: referente a qual projeto a ata de reunião será aberta;
b) `plano`: referente a qual plano a ata de reunião será aberta;
c) `descrição` da ata: texto resumido a respeito do que se tratou a reunião;
d) `assunto`: quais assuntos serão tratados na reunião. Os assuntos devem corresponder ao plano a qual a ata foi aberta;
e) `decisões`: engloba as decisões tomadas durante a reunião;
f) `participantes`: lista os nomes de todas as pessoas que participaram da reunião, podendo ser relacionadas pessoas externas ao projeto;
g) `data da reunião`: quando a reunião foi realizada.

3.3.2.7 Cadastro de risco

Com a equipe e o projeto definido, na reunião inicial, os integrantes podem preencher os riscos e adicioná-los no plano do projeto. Conforme o guia de implementação do MPT.BR, risco é o artefato no qual mantém-se os registros de todos os riscos encontrados no projeto. Este artefato é um item dentro do plano de testes ou de aceitação. É comum deixar as

informações do risco na mesma tela do cadastro de projetos, mas para esse, foi escolhido desenvolver uma nova tela, para uma melhor visualização das informações. Na figura 18 é possível visualizar a tela de risco.

Figura 18 – Interface Risco



Os campos do cadastro de risco são:

a) `título`: representa uma identificação para o risco;
b) `descrição`: representa um detalhamento do risco;
c) `responsável`: representa a pessoa responsável por monitorar e tratar o risco;
d) `impacto`: representa a proporção do risco perante o projeto;
e) `probabilidade`: representa um percentual de que o risco possa acontecer;
f) `tipo de risco`: representa o tipo de risco, podendo ser risco de projeto ou risco de produto;
g) `mitigação`: representa a descrição do plano de mitigação para o risco;
h) `contingência`: representa a descrição do plano de contingência a ser adotado.

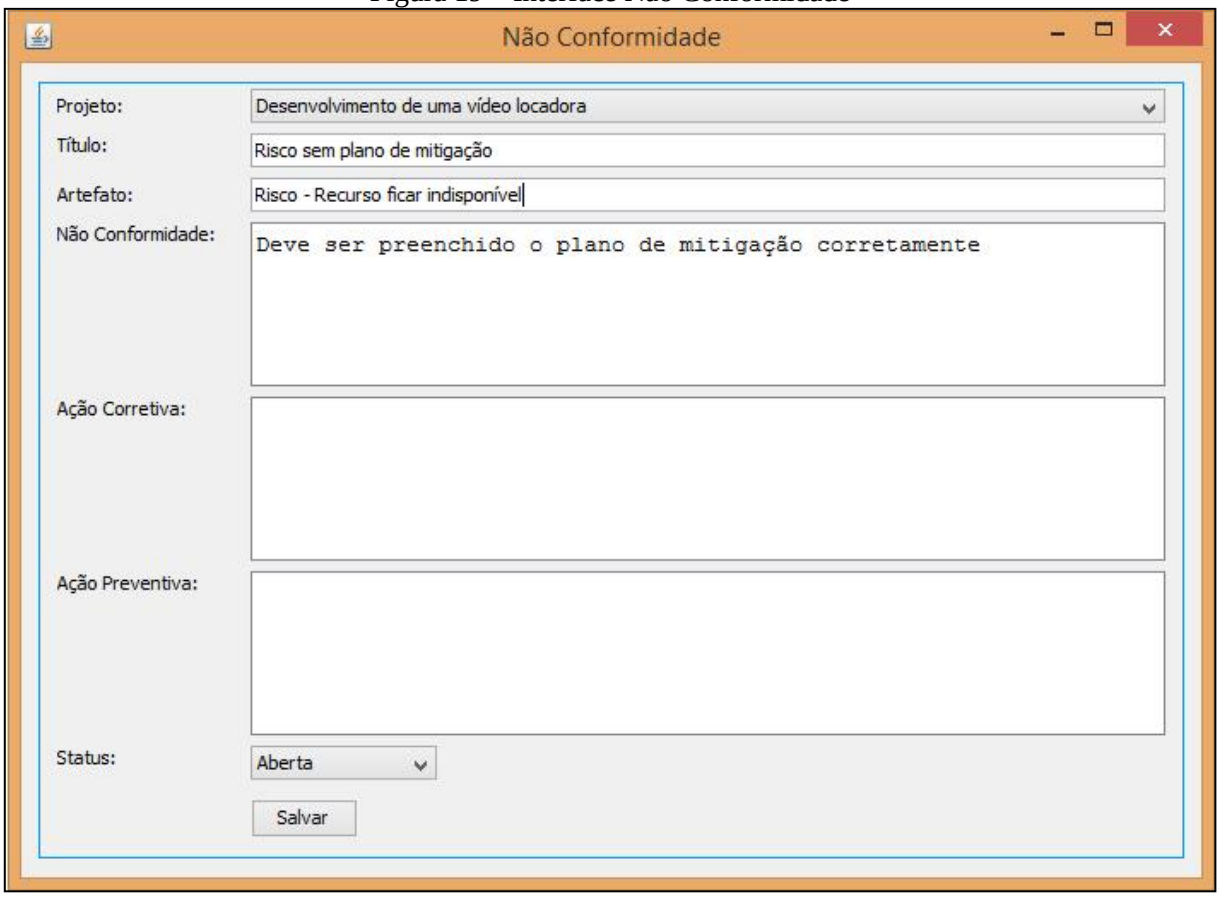
Como o risco está diretamente ligado ao plano e ao projeto, deve-se escolher qual plano e projeto serão anexados ao risco.

Quando o risco estiver devidamente cadastrado, clica-se em salvar para gravar o risco.

3.3.2.8 Cadastrar não conformidade

Segundo o guia de implementação do MPT.BR, não conformidades são utilizadas para registrar inconsistências entre os artefatos cadastrados no projeto e o processo MPT.BR. Caso algum artefato esteja fora das práticas do processo deve-se registrar uma não conformidade para que um usuário possa tratar essa inconsistência. Na figura 19 tem-se uma visão da tela de cadastro de não conformidade.

Figura 19 – Interface Não Conformidade



Os campos para cadastro são:

a) `título`: representa uma descrição detalhada para a abertura de uma não conformidade;
b) `artefato`: representa o artefato que está em desacordo com o projeto;
c) `não conformidade`: explicação do erro encontrado no artefato;
d) `ação corretiva`: que ação deve ser tomada para corrigir o problema;
e) `ação preventiva`: o que será feito para que o problema não volte a acontecer;
f) `status`: aberta, caso a não conformidade ainda não tenha sido tratada, e fechada caso tenha sido resolvida.

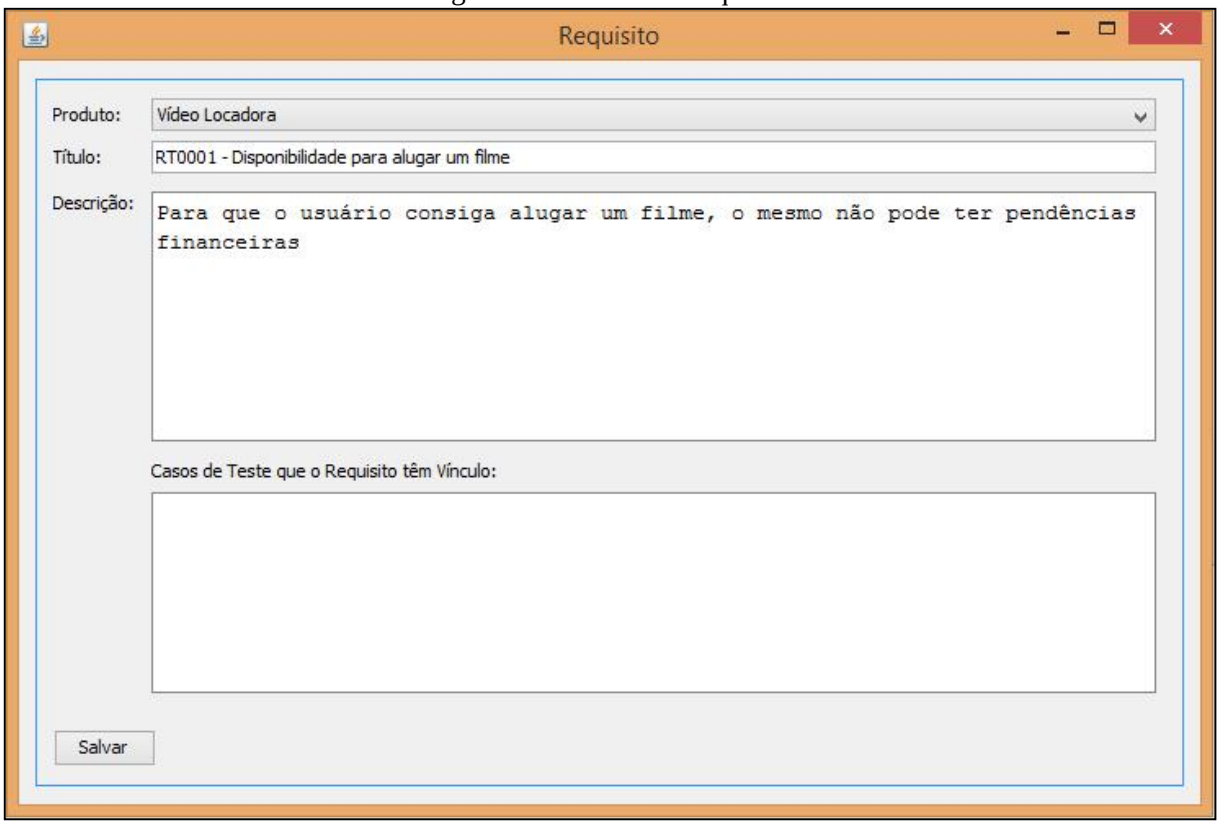
A não conformidade é anexada diretamente no projeto, sendo assim, é possível escolher em qual projeto ela fará parte.

Após a edição da não conformidade, clica-se em salvar para efetuar as alterações.

3.3.2.9 Cadastro de requisito

O requisito de teste é o artefato que traça a criação de casos de teste. Com ele sabe-se quais funcionalidades do software devem ser testadas. Na figura 20 é possível visualizar a interface do cadastro de requisito.

Figura 20 – Interface Requisito



Os campos para cadastro são:

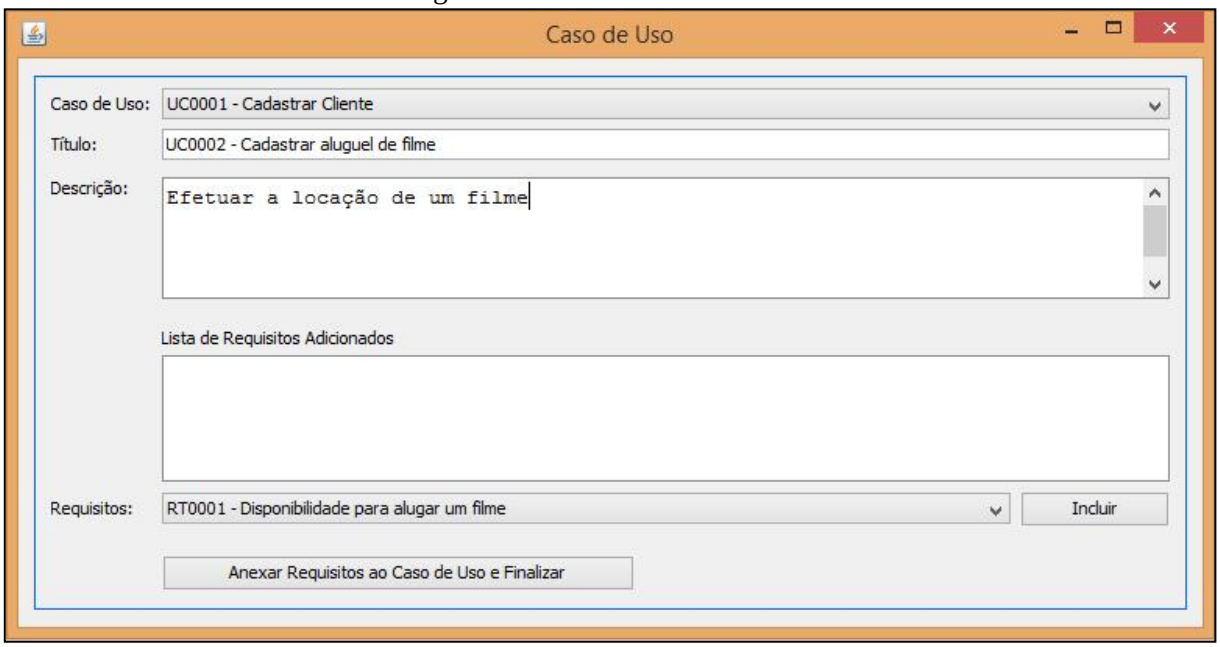
a) `produto`: este artefato é ligado ao produto e não ao projeto; sendo assim, é possível criar outros projetos aos quais o requisito estará vinculado;
b) `título`: um identificador do requisito;
c) `descrição`: um detalhamento do artefato.

Após efetuar o cadastro do artefato, clica-se em salvar para gravar as alterações.

3.3.2.10 Cadastro de caso de uso

O artefato de caso de uso é o predecessor do caso de teste. Ele funciona como um agrupador de requisitos. Na figura 21 é possível visualizar a tela de seu cadastro.

Figura 21 – Interface Caso de Uso



Os campos para cadastro são:

a) `título`: uma breve descrição do caso de uso;

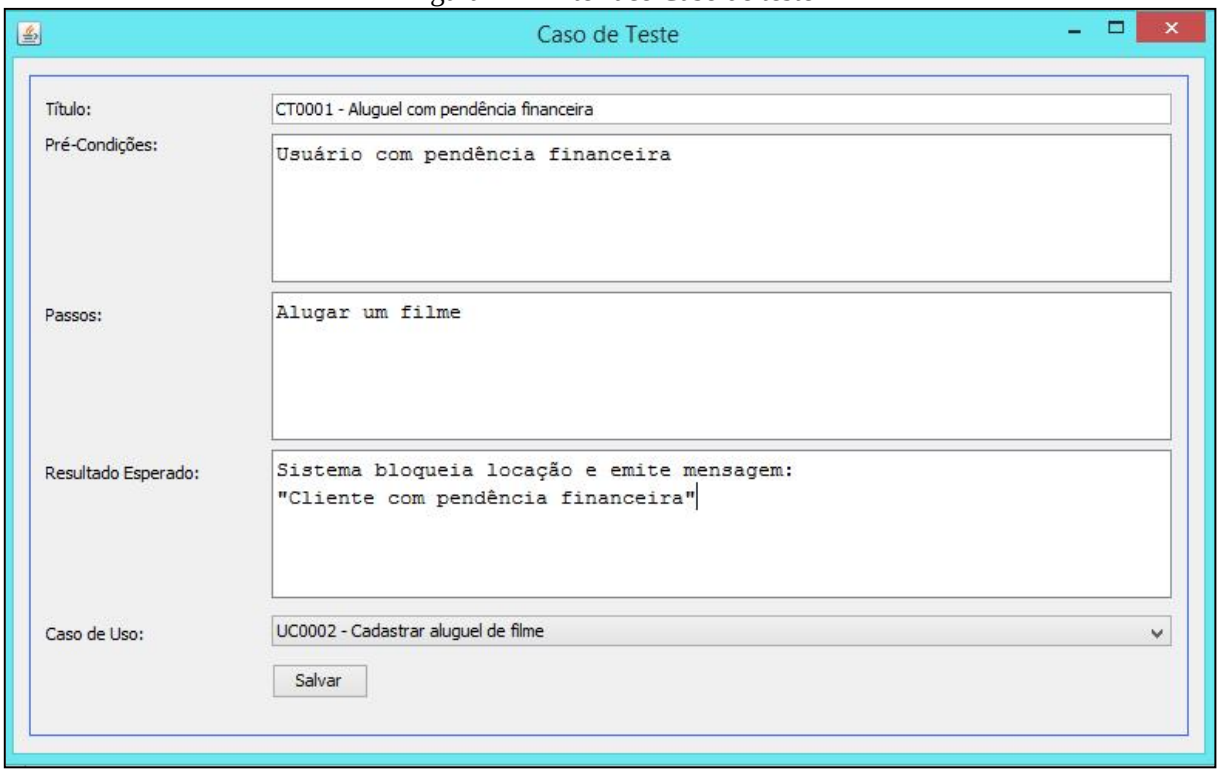
b) `descrição`: detalhamento do artefato caso de uso.

No campo requisitos, são listados todos os requisitos cadastrados até o momento. Cabe ao usuário atribuir quais requisitos farão parte do caso de uso. Para isso, basta escolher o requisito e clicar no botão incluir. Após o término do cadastro e de anexar os requisitos, clica-se em anexar requisitos ao caso de uso e finalizar, para a conclusão do caso de uso.

3.3.2.11 Caso de teste

Caso de teste é um artefato que está disponível a partir do nível 2 do MPT.BR. Neste artefato, são informados os testes que serão efetuados em determinada funcionalidade do sistema. Tem vínculo direto com o caso de uso, pois é a partir do caso de uso que devem ser projetados os casos de testes necessários. Na figura 22 é possível visualizar a interface.

Figura 22 – Interface Caso de teste



Os campos para cadastro são:

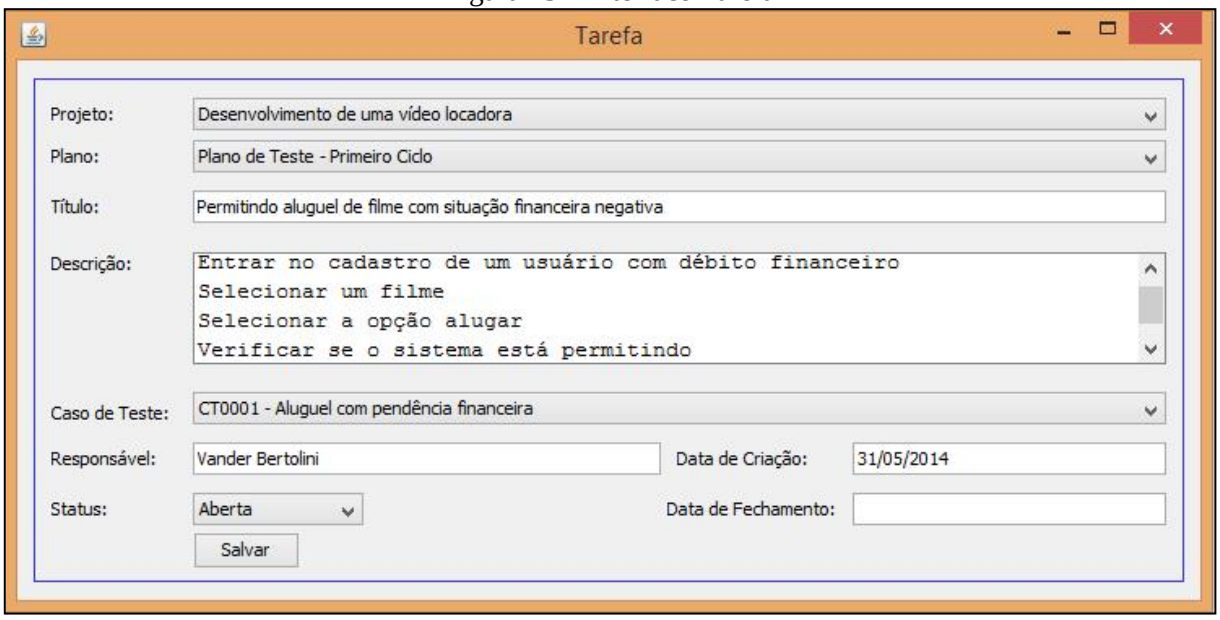
a) `título`: um identificador para o caso de teste;
b) `pré-condição`: o estado que deve estar a aplicação testada para o início da execução do caso de teste;
c) `passos`: o que é necessário, para acontecer o que foi proposto no caso de teste;
d) `resultado esperado`: o resultado esperado após a execução do caso de teste.

Por fim liga-se o caso de uso que terá vínculo com o caso de teste e em seguida clica-se em salvar para efetuar as alterações.

3.3.2.12 Cadastrar tarefa

Tarefa é o artefato no qual registra-se os acontecimentos das atividades do projeto. Nela são registrados vários pontos como sugestões, implementações do produto, atividades a serem desenvolvidas (como por exemplo os casos de teste que têm papel fundamental na tarefa, pois é nela que o caso de teste é anexado). Na figura 23 é possível visualizá-la.

Figura 23 – Interface Tarefa



Os campos para cadastro são:

a) `título`: descrição breve sobre o que se trata a tarefa;
b) `descrição`: detalhar todas as atividades a serem executadas na tarefa;
c) `responsável`: pessoa responsável por executar as atividades da tarefa;
d) `data de criação`: data em que a tarefa foi registrada;
e) `status`: aberta, se a tarefa está sendo executada, e fechada, se todas as atividades foram concluídas.

A tarefa tem ligação direta com o plano e projeto. Sendo assim, deve-se escolher em qual plano e projeto a tarefa fará parte, em seguida deve-se também escolher a qual caso de teste a tarefa fará parte. Por fim, clica-se em salvar para efetuar as alterações.

3.3.2.13 Relatórios

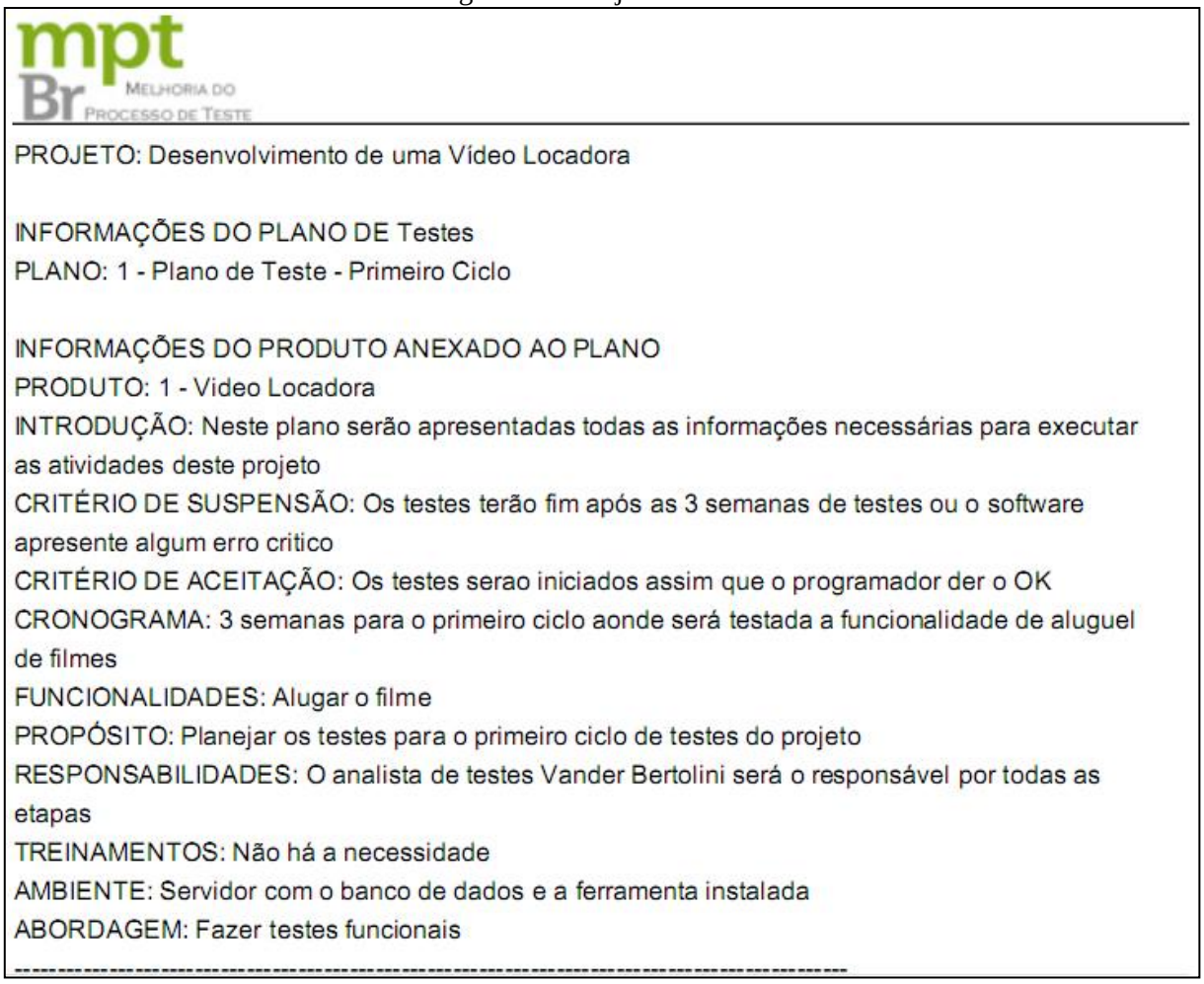
Para atender aos casos de uso `UC16 – Visualizar Relatório do Plano de Teste`, `UC15 – Visualizar Relatório do Plano de Aceitação`, `UC14 – Visualizar Relatório da Ata de Reunião` e `UC13 – Visualizar Relatório das Tarefas do Plano`, foi desenvolvida a tela de relatórios, na qual o usuário pode selecionar qual relatório deverá ser gerado. Na figura 24 é possível visualizar esta tela.

Figura 24 – Interface Relatórios MPT

Para a construção dos relatórios foi utilizada a biblioteca gratuita `itext`, que possui código fonte aberto e é exclusiva para a linguagem Java.

O primeiro relatório, chamado de projetos abertos, lista informações do projeto e dos planos anexados ao projeto. Na figura 25 é possível visualizar o relatório de projetos.

Figura 25 – Projetos Abertos


mpt Br Melhoria do Processo de Teste

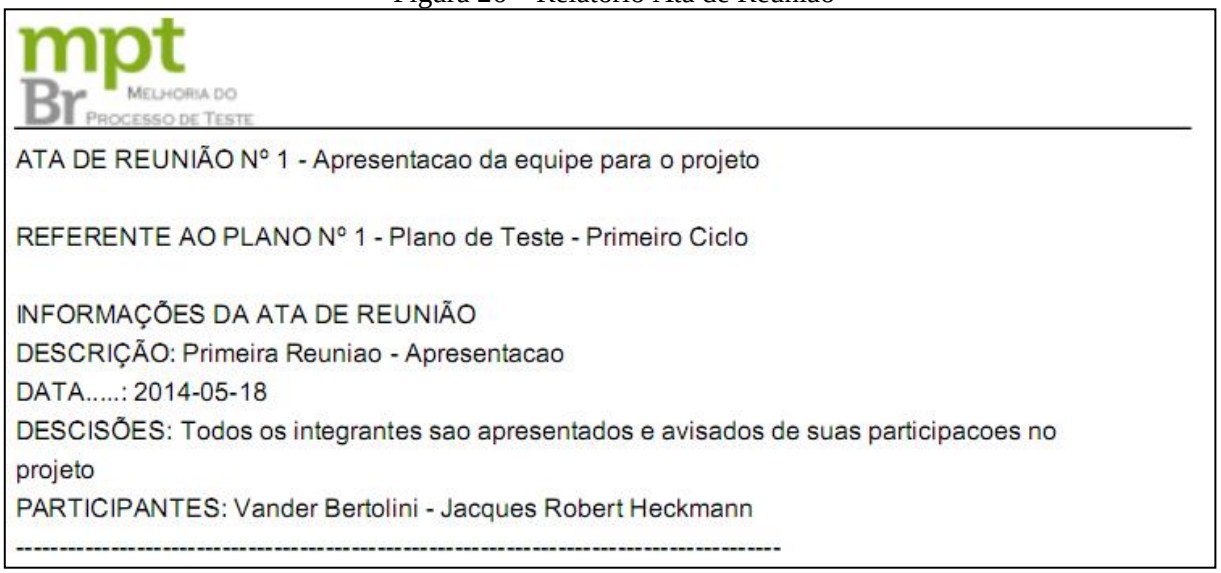
PROJETO: Desenvolvimento de uma Vídeo Locadora

INFORMAÇÕES DO PLANO DE Testes
PLANO: 1 - Plano de Teste - Primeiro Ciclo

INFORMAÇÕES DO PRODUTO ANEXADO AO PLANO
PRODUTO: 1 - Video Locadora
INTRODUÇÃO: Neste plano serão apresentadas todas as informações necessárias para executar as atividades deste projeto
CRITÉRIO DE SUSPENSÃO: Os testes terão fim após as 3 semanas de testes ou o software apresente algum erro critico
CRITÉRIO DE ACEITAÇÃO: Os testes serao iniciados assim que o programador der o OK
CRONOGRAMA: 3 semanas para o primeiro ciclo aonde será testada a funcionalidade de aluguel de filmes
FUNCIONALIDADES: Alugar o filme
PROPÓSITO: Planejar os testes para o primeiro ciclo de testes do projeto
RESPONSABILIDADES: O analista de testes Vander Bertolini será o responsável por todas as etapas
TREINAMENTOS: Não há a necessidade
AMBIENTE: Servidor com o banco de dados e a ferramenta instalada
ABORDAGEM: Fazer testes funcionais

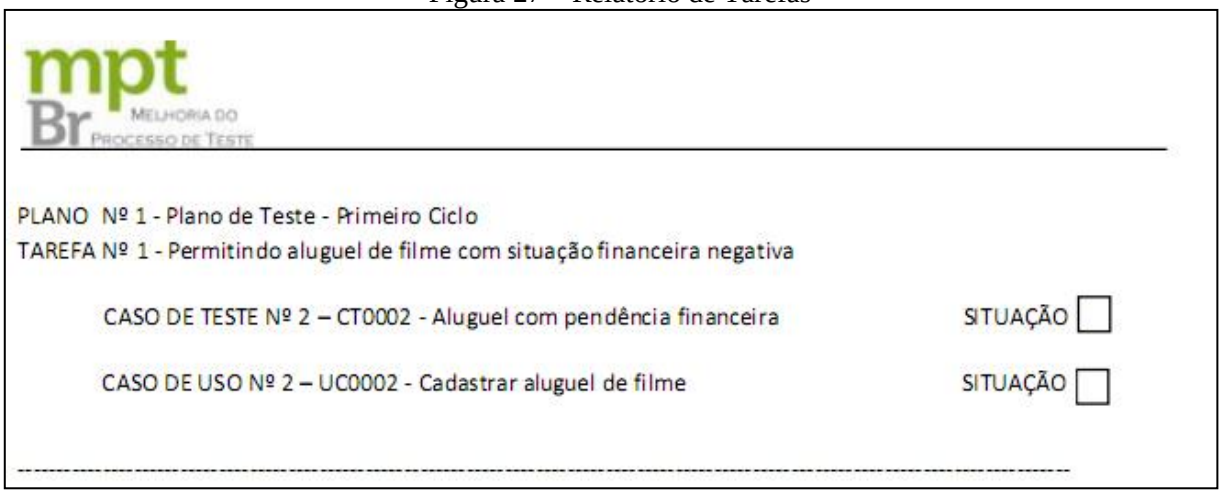
O segundo relatório, chamado de atas de reunião, lista todas as reuniões realizadas para cada plano aberto. Serve para que o usuário possa ter documentado todas as discussões realizadas para um projeto aberto, segundo o manual de implementação do MPT.BR. Na figura 26 é possível visualizar o relatório.

Figura 26 – Relatório Ata de Reunião



Por fim, tem-se o relatório de tarefas, que serve como um *checklist* para que o usuário saiba o que já foi feito e o que ainda falta fazer. Nele são listadas as tarefas com seus respectivos artefatos criados, sendo que para cada artefato é ilustrada a situação como concluída ou não concluída. Na figura 27 é possível visualizar o relatório de tarefas.

Figura 27 – Relatório de Tarefas



3.4 RESULTADOS E DISCUSSÃO

A proposta inicial deste trabalho era desenvolver uma ferramenta que apoiasse o gerenciamento de projetos de teste do modelo MPT.BR no nível 3 de um total de 5 níveis. Também era proposta uma revisão nos níveis de maturidade 1 e 2 desenvolvidos por Amorim (2011). No entanto, ao utilizar o projeto anteriormente desenvolvido viu-se que a ferramenta Visual Editor (utilizada naquela versão), foi descontinuada. Desta forma foi necessário um novo planejamento para que o projeto anterior conseguisse ser executado.

Algumas práticas do nível 3 propostas não foram implantadas, pois observou-se que não cabiam à ferramenta, como por exemplo as práticas “preparar ambiente para aceitação”, que consiste em o usuário disponibilizar ferramentas para a execução do teste, “conduzir testes de aceitação”, que consiste em gerar um relatório sobre as ferramentas utilizadas nos teste, e “avaliar condições de aceitação”, que deve ser executada em nível de cliente (ou seja, o contratante do software faz uma avaliação se a ferramenta está dentro do que foi ofertado pela empresa contratada).

Os resultados obtidos no término do desenvolvimento da ferramenta são satisfatórios, e permitem apoiar o gerenciamento de projetos de teste. Ao realizar uma comparação com a ferramenta desenvolvida e os trabalhos correlatos, pode-se concluir que a ferramenta atende às atividades do planejamento de testes, baseando-se no padrão de documentação IEEE-829.

Foi feito um estudo sobre o ambiente PDE. Constatou-se que é um ambiente que ainda tem muito para evoluir. Da versão criada em 2011 por Amorim (2011), com a utilizada na construção desta ferramenta, viu-se que ainda não há uma interface rica e tudo deve ser criado manualmente. Bibliotecas auxiliares como o *driver* JDBC ainda causam problemas de compatibilidade, sendo necessário criar um novo *plug-in* que encapsule o *driver* e depois utilize-o como um componente na utilização do *plug-in* principal.

No quadro 4 faz-se uma comparação entre os trabalhos correlatos e a ferramenta desenvolvida, sobre os processos do nível 3.

Quadro 4 – Comparação entre trabalhos correlatos nível 3

Área	**Ferramenta**		
	XPlanner	**Meisen (2005)**	**Ferramenta Desenvolvida**
Fechamento do Teste (FDT)	Atende	Atende parcialmente	Atende parcialmente
Medição e Análise do Teste (MAT)	Atende	Atende	Atende
Teste de Aceitação (TDA)	Atende parcialmente	Atende parcialmente	Atende
Gerência de Projetos de Teste (GPT)	Atende	Atende parcialmente	Atende

Na primeira versão do trabalho que contemplava os níveis de maturidade 1 e 2 do modelo MPT.BR algumas áreas eram atendidas de forma parcial. Como o nível 3 é um complemento de algumas áreas dos níveis 1 e 2, ao se realizar uma nova análise com os trabalhos correlatos, conclui-se que a ferramenta as atende por completo, conforme o quadro 5.

Quadro 5 – Comparação entre trabalhos correlatos nível 1 e 2

Área	**Ferramenta**		
	XPlanner	**Meisen (2005)**	**Ferramenta desenvolvida**
Gerência de Projetos de Teste (GPT)	Atende	Atende parcialmente	Atende
Projeto e Execução de Teste (PET)	Não atende	Atende parcialmente	Atende
Gerência de Requisitos de Teste (GRT)	Atende	Atende	Atende

Na comparação observa-se que o XPlanner é o trabalho correlato que mais se identifica com a ferramenta desenvolvida, pelo fato de ambas serem voltadas para a área de teste de software.

4 CONCLUSÕES

O objetivo da ferramenta desenvolvida foi dar continuidade ao projeto de Amorim (2011), atingindo o nível 3 de maturidade chamado de “definido” do modelo MPT.BR. Anteriormente ao desenvolvimento desta ferramenta, o nível 1 e 2 no trabalho anterior eram atendidos de forma parcial, pois algumas áreas do MPT.BR se estendem por mais de um nível. Com o término desta, ficam concluídos os três níveis de maturidade, dentre um total de cinco abordados pelo modelo com exceção das práticas não implementadas da área Fechamento do Teste conforme descrito na seção 3.4.

Como a ferramenta é uma continuação de um trabalho anterior, investiu-se um grande tempo no entendimento desse. A biblioteca `EditorPart` que era grande foco do trabalho anterior para a criação de interfaces gráficas, foi descontinuada. Sendo assim, foram gastos quase dois meses para adaptar toda a ferramenta sem utilizar a biblioteca `EditorPart`.

O recurso de criação de *plug-ins* do Eclipse obteve uma significativa melhoria, no quesito de compatibilidade com banco de dados, que, segundo Amorim (2011), era muito limitado para lidar com recursos externos que não eram diretamente ligados ao Eclipse.

A utilização do MySQL como banco de dados foi muito satisfatória, pois a ferramenta desenvolvida não exige grande performance, o ambiente é totalmente gratuito e possui código fonte aberto. Sendo assim, faz-se com que o MySQL seja a principal escolha para a criação desta ferramenta.

A ferramenta em forma de *plug-in* pode contribuir muito em um ambiente de desenvolvimento, pois o usuário tem em uma única ferramenta, um ambiente de teste completo que atenda a normas de qualidade de software, disponibilizando toda uma estrutura de criação de planos de testes, requisitos, casos de uso e casos de teste que são fundamentais para garantir a qualidade do software a ser desenvolvido. Pelo fato de a ferramenta ser gratuita e de código fonte aberto, possibilita que qualquer desenvolvedor possa utilizá-la, bastando apenas ter o ambiente Eclipse configurado.

Pelo fato da ferramenta ser um *plug-in*, ela apenas irá agregar ao ambiente Eclipse, já que vai funcionar como uma opção a mais no ambiente de desenvolvimento e sem a necessidade de se ter ferramentas auxiliares.

O desenvolvimento da ferramenta foi uma experiência significante, pois pode-se estudar a fundo o modelo MPT.BR, que age de uma maneira à parte se comparado com outros modelos de teste, pois este trabalha como um projeto em separado, sem a necessidade de se

estar acoplado em algum projeto específico. Possibilitou-se também adquirir experiência na criação de *plug-ins* através da utilização do recurso PDE do Eclipse.

4.1 EXTENSÕES

Como sugestão de extensões futuras para este trabalho, pode-se implementar artefatos baseados nas práticas dos níveis 4 e 5 do modelo de teste MPT.BR.

Outra sugestão é permitir a importação de dados já existentes em planilhas. Isto pode ser bastante útil quando o interessado na ferramenta já utilizar planilhas para fazer o planejamento e análise de seus testes, podendo-se então utilizá-las para importar casos de teste, casos de uso e requisitos.

Por fim, tornar possível anexar evidências e documentos, para o complemento e entendimento do artefato de teste criado.

REFERÊNCIAS

ALECRIM, Emerson. **Banco de dados MySQL e PostgreSQL**. [S.1], [2008]. Disponível em:< http://www.infowester.com/postgremysql.php>. Acesso em: 25 maio 2014.

AMORIM, Daniel R. **Ferramenta de apoio a implementação do processo melhoria de processo de processo de teste (MPT).** 2011. 91 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

ASSOCIAÇÃO LATINO AMERICANA DE TESTE DE SOFTWARE. **MPT.Br:** melhoria do processo de teste de software brasileiro - introdução ao modelo. [Rio de Janeiro], [2002]. Disponível em: <http://www.alats.org.br/Default.aspx?tabid=252>. Acesso em: 02 jun. 2013.

BASTOS, Anderson et al. **Base de conhecimento em teste de software.** Rio de Janeiro: Martins, 2007.

COUTO, Ana B. **CMMI**: integração dos modelos de capacitação e maturidade de sistemas. Rio de Janeiro: Ciência Moderna, 2007.

ECLIPSE FOUNDATION. **PDE**. [S.l.], [2010]. Disponível em: <http://www.eclipse.org/pde>. Acesso em: 03 jun. 2013.

HUTTERS, Thomas. **Why MySQL?** [S.1], [2010]. Disponível em: <http://www.mysql.com/why-mysql/scaleout/thephonehouse.html>. Acesso em: 1 jun. 2014.

KLAIS SOLUÇÕES CONSULTORIA E DESENVOLVIMENTO LTDA. **Automação de processos**: simplificando e diminuindo custos. [Campinas], 2008. Disponível em: <http://www.klais.com.br/automacao.html>. Acesso em: 02 jun. 2013.

MEISEN, Mariane. **Ferramenta de apoio a gerencia de requisitos baseado no modelo CMMI.** 2005. 87 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau. Disponível em: <http://www.bc.furb.br/docs/mo/2005/307149_1_1.pdf>. Acesso em: 02 jun. 2013.

MIGUEL, Sérgio B. **Padrão para documentação de teste de software**. [S.1], [2014?]. Disponível em: < http://www.devmedia.com.br/padrao-para-documentacao-de-teste-de-software/26534>. Acesso em: 05 abr. 2014.

OLIVEIRA, Fabricio. **Software de apoio à gerência de solicitação de mudanças.** 2006. 72 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

PEDROSA, Ana V. et al. **Uso de XML em uma IDE: uma explanação do modelo de plug-ins da plataforma Eclipse**. [S.1], [2014?]. Disponível em: <http://www.dei.unicap.br/~almir/seminarios/2004.2/ts04/xmlide/conceitosdeambienteintegrado.html>. Acesso em: 01 jun. 2014.

PEZZÈ, Mauro; YOUNG, Michal. **Teste e análise de software.** Tradução Bernardo Copstein, Flavio Moreira de Oliveira. Porto Alegre: Bookman, 2008.

RIOS, Emerson. **Documentação de teste de software:** dissecando o padrão IEEE-829. Niterói: Imagem Art Studio, 2007.

ROCHA, André D. **Artigo java magazine 37 – criando plug-ins para o Eclipse**. [S.1], [2014?]. Disponível em: < http://www.devmedia.com.br/artigo-java-magazine-37-criando-plug-ins-para-o-eclipse/8942>. Acesso em: 01 jun. 2014.

SERENA SOFTWARE INCORPORATION. **Serena open source and hosted Project management software.** [Redwood City], [2008]. Disponível em: <http://openproj.org>. Acesso em: 14 abr. 2013.

SOFTEX RECIFE. **MPT.Br.** [Recife], 2013a. Disponível em: <http://www.mpt.org.br/mptbr/>. Acesso em: 1 mar. 2013.

_____. **MPT.BR:** melhoria do processo de teste de brasileiro - guias. [Recife], 2013b. Disponível em: <http://www.mpt.org.br/guias/>. Acesso em: 20 mar. 2013.

_____**.MPT.BR:** melhoria do processo de teste brasileiro – como certificar sua empresa. [Recife], 2014c. Disponível em: < http://mpt.org.br/mpt/empresas-certificadas/como-certificar-sua-empresa/>. Acesso em: 01 mar. 2014.

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

XPLANNER. **Overview**. [S.l.], [2002?]. Disponível em: <http://www.xplanner.org>. Acesso em: 02 jun. 2013.

APÊNDICE A – Relação dos casos de uso da ferramenta

São apresentados nos quadros 6 a 22 o detalhamento dos casos de uso apresentados na seção 3.2.1.

Quadro 6 - UC01 Cadastrar Usuários e Permissões de Acesso

Nome	Cenários	Etapas
UC01 – Cadastrar Usuários e Permissões de Acesso	Cadastro com sucesso	1 – O usuário informa os dados do novo usuário da ferramenta 2 – O usuário cadastras as permissões de acesso 3 – O usuário salva as alterações 4 – O sistema valida se todos os campos foram informados 5 – O sistema salva o usuário
	Nome de usuário não informado	1 – O usuário deixa em branco o nome de usuário 2 – O usuário salva as informações 3 – O sistema valida se o nome foi informado 4 – O sistema emite a mensagem de que o nome de usuário é obrigatório
	Senha não informada	1 – O usuário deixa em branco a senha 2 – O usuário salva as informações 3 – O sistema valida as informações 4 – O sistema emite a mensagem de que a senha é obrigatória

Quadro 7 – UC02 Cadastro de cargos

Nome	Cenários	Etapas
UC02 – Cadastrar cargos	Cadastro com sucesso	1 – O usuário informa a descrição do cargo 2 – O usuário salva as alterações 3 – O sistema valida se todos os campos foram informados 4 – O sistema salva as alterações
	Descrição do cargo não informada	1 - O usuário deixa em branco a descrição do cargo 2 – O usuário salva as alterações 3 – O sistema valida as informações 4 – O sistema emite mensagem de erro de que a descrição do cargo é obrigatória

Quadro 8 – UC03 Cadastrar caso de teste

Nome	Cenários	Etapas
UC03 Cadastrar caso de teste	Cadastro com sucesso	1 – O usuário informa os campos apresentados na tela 2 – O usuário salva as informações 3 – O sistema valida as informações 4 – O sistema salva as informações
	Título não informado	1 – O usuário deixa em branco a descrição do título do caso de teste 2 – O usuário salva as informações 3 – O sistema valida as informações 4 – O sistema emite a mensagem de erro de que o título é obrigatório

Quadro 9 – UC04 Cadastrar projeto

Nome	Cenários	Etapas
UC04 Cadastrar projeto	Cadastro com sucesso	1 - O usuário informa os campos apresentados na tela 2 – O usuário salva as informações 3 – O sistema valida as informações 4 – O sistema salva as

	Descrição do projeto não informada	informações 1 – O usuário deixa em branco o campo descrição do projeto 2 – O usuário salva as informações 3 – O sistema valida as informações 4 – O sistema emite mensagem de que a descrição do projeto deve ser informada

Quadro 10 – UC05 Cadastrar requisitos

Nome	Cenários	Etapas
UC05 Cadastrar requisitos	Cadastro com sucesso	1 - O usuário informa os campos apresentados na tela 2 – O usuário salva as informações 3 – O sistema valida as informações 4 – O sistema salva as informações
	Título não informado	1 – O usuário deixa em branco o campo título 2 – O usuário salva as informações 3 – O sistema valida as informações 4 – O sistema emite mensagem de que o título deve ser informado

Quadro 11 – UC06 Cadastrar produto

Nome	Cenários	Etapas
UC06 Cadastrar produto	Cadastro com sucesso	1 - O usuário informa os campos apresentados na tela 2 – O usuário salva as informações 3 – O sistema valida as informações 4 – O sistema salva as informações
	Descrição do produto não informada	1 – O usuário deixa em branco o campo descrição 2 – O usuário salva as informações 3 – O sistema valida as informações 4 – O sistema emite mensagem de que a descrição

		deve ser informada

Quadro 12 – UC07 Cadastrar riscos

Nome	Cenários	Etapas
UC07 Cadastrar riscos	Cadastro com sucesso	1 - O usuário informa os campos apresentados na tela 2 – O usuário salva as informações 3 – O sistema valida as informações 4 – O sistema salva as informações
	Título não informado	1 – O usuário deixa em branco o campo título 2 – O usuário salva as informações 3 – O sistema valida as informações 4 – O sistema emite mensagem de que o título deve ser informado
	Projeto não informado	1 – O usuário deixa em branco o campo projeto 2 – O usuário salva as informações 3 – O sistema valida as informações 4 – O sistema emite mensagem de que o projeto deve ser informado
	Plano não informado	1 – O usuário deixa em branco o campo plano 2 – O usuário salva as informações 3 – O sistema valida as informações 4 – O sistema emite mensagem de que o plano deve ser informado
	Responsável não informado	1 – O usuário deixa em branco o campo responsável 2 – O usuário salva as informações 3 – O sistema valida as informações 4 – O sistema emite mensagem de que o responsável deve ser

		informado

Quadro 13 – UC8 Cadastrar plano de teste

Nome	Cenários	Etapas
UC8 Cadastrar plano de teste	Cadastro com sucesso	1 - O usuário informa os campos apresentados na tela 2 – O usuário salva as informações 3 – O sistema valida as informações 4 – O sistema salva as informações
	Título não informado	1 – O usuário deixa em branco o campo título 2 – O usuário salva as informações 3 – O sistema valida as informações 4 – O sistema emite mensagem de que o título deve ser informado

Quadro 14 – UC9 Cadastrar tarefa

Nome	Cenários	Etapas
UC9 Cadastrar tarefa	Cadastro com sucesso	1 - O usuário informa os campos apresentados na tela 2 – O usuário salva as informações 3 – O sistema valida as informações 4 – O sistema salva as informações
	Título não informado	1 – O usuário deixa em branco o campo título 2 – O usuário salva as informações 3 – O sistema valida as informações 4 – O sistema emite mensagem de que o título deve ser informado
	Projeto não informado	1 – O usuário deixa em branco o campo projeto 2 – O usuário salva as informações 3 – O sistema valida as informações 4 – O sistema emite mensagem de que o projeto deve ser informado

	Plano não informado	1 – O usuário deixa em branco o campo plano 2 – O usuário salva as informações 3 – O sistema valida as informações 4 – O sistema emite mensagem de que o plano deve ser informado
	Caso de teste não informado	1 – O usuário deixa em branco o campo caso de teste 2 – O usuário salva as informações 3 – O sistema valida as informações 4 – O sistema emite mensagem de que o caso de teste deve ser informado
	Responsável não informado	1 – O usuário deixa em branco o campo responsável 2 – O usuário salva as informações 3 – O sistema valida as informações 4 – O sistema emite mensagem de que o responsável deve ser informado

Quadro 15 – UC10 Cadastrar caso de uso

Nome	**Cenários**	**Etapas**
UC10 Cadastrar caso de uso	Cadastro com sucesso	1 - O usuário informa os campos apresentados na tela 2 – O usuário salva as informações 3 – O sistema valida as informações 4 – O sistema salva as informações
	Título não informado	1 – O usuário deixa em branco o campo título 2 – O usuário salva as informações 3 – O sistema valida as informações 4 – O sistema emite mensagem de que o título deve ser informado
	Requisito não informado	1 – O usuário deixa em branco o campo requisito 2 – O usuário salva as

		informações 3 – O sistema valida as informações 4 – O sistema emite mensagem de que o requisito deve ser informado

Quadro 16 – UC11 Cadastrar ata de reunião

Nome	Cenários	Etapas
UC11 Cadastrar ata de reunião	Cadastro com sucesso	1 - O usuário informa os campos apresentados na tela 2 – O usuário salva as informações 3 – O sistema valida as informações 4 – O sistema salva as informações
	Descrição da ata não informada	1 – O usuário deixa em branco o campo descrição da ata 2 – O usuário salva as informações 3 – O sistema valida as informações 4 – O sistema emite mensagem de que a descrição da ata deve ser informada
	Projeto não informado	1 – O usuário deixa em branco o campo projeto 2 – O usuário salva as informações 3 – O sistema valida as informações 4 – O sistema emite mensagem de que o projeto deve ser informado
	Plano não informado	1 – O usuário deixa em branco o campo plano 2 – O usuário salva as informações 3 – O sistema valida as informações 4 – O sistema emite mensagem de que o plano deve ser informado

Quadro 17 – UC12 Cadastrar não conformidade

Nome	Cenários	Etapas
UC12 Cadastrar não conformidade	Cadastro com sucesso	1 - O usuário informa os campos apresentados na tela 2 – O usuário salva as

		informações 3 – O sistema valida as informações 4 – O sistema salva as informações
	Título não cadastrado	1 – O usuário deixa em branco o campo título 2 – O usuário salva as informações 3 – O sistema valida as informações 4 – O sistema emite mensagem de que o título deve ser informado
	Artefato não cadastrado	1 – O usuário deixa em branco o campo artefato 2 – O usuário salva as informações 3 – O sistema valida as informações 4 – O sistema emite mensagem de que o artefato deve ser informado
	Projeto não informado	1 – O usuário deixa em branco o campo projeto 2 – O usuário salva as informações 3 – O sistema valida as informações 4 – O sistema emite mensagem de que o projeto deve ser informado

Quadro 18 – UC17 Extrair indicadores

Nome	Cenários	Etapas
UC17 Extrair indicadores	Visualização com sucesso	1 – Usuário clica no botão atualizar indicadores 2 – Sistema valida informação 3 – Sistema atualiza os valores atuais

Quadro 19 – UC16 Visualizar relatório do plano de teste

Nome	Cenários	Etapas
UC16 Visualizar relatório do plano de teste	Visualização com sucesso	1 – Usuário seleciona a opção de relatório de plano 2 – Sistema verifica se existe algum plano de teste 3 – Sistema emite relatório

Quadro 20 – UC15 Visualizar relatório do plano de aceitação

Nome	Cenários	Etapas
UC15 Visualizar relatório do plano de aceitação	Visualização com sucesso	1 – Usuário seleciona a opção de relatório de plano 2 – Sistema verifica se existe algum plano de aceitação 3 – Sistema emite relatório

Quadro 21 – UC14 Visualizar relatório da ata de reunião

Nome	Cenários	Etapas
UC14 Visualizar relatório da ata de reunião	Visualização com sucesso	1 – Usuário seleciona a opção de relatório de ata de reunião 2 – Sistema verifica se existe alguma ata de reunião 3 – Sistema emite relatório

Quadro 22 – UC13 Visualizar relatório das tarefas do plano

Nome	Cenários	Etapas
UC13 Visualizar relatório das tarefas do plano	Visualização com sucesso	1 – Usuário seleciona a opção de relatório de tarefas do plano 2 – Sistema verifica se existe alguma tarefa 3 – Sistema emite relatório