UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE SISTEMAS DE INFORMAÇÃO – BACHARELADO

FERRAMENTA CASE PARA O PROCESSO DE MANUTENÇÃO DE SOFTWARE BASEADO NA NORMA ISO/IEC 14764

LEONARDO RAFAEL WEHRMEISTER

BLUMENAU
2004

2004/2-06

LEONARDO RAFAEL WEHRMEISTER

FERRAMENTA CASE PARA O PROCESSO DE MANUTENÇÃO DE SOFTWARE BASEADO NA NORMA ISO/IEC 14764

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso do curso de Sistemas de Informação – Bacharelado.

Prof. Fabiane Barreto Vavassori Benitti – Orientador

BLUMENAU
2004

2004/2-06

FERRAMENTA CASE PARA O PROCESSO DE MANUTENÇÃO DE SOFTWARE BASEADO NA NORMA ISO/IEC 14764

Por

LEONARDO RAFAEL WEHRMEISTER

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

Presidente:	Prof. Fabiane Barreto Vavassori Benitti – Orientador, FURB
Membro:	Prof. Marcel Hugo, FURB
Membro:	Prof. Luiz Bianchi, FURB

Blumenau, 24 de fevereiro de 2005

Dedico este trabalho a todos os amigos, especialmente aqueles que me ajudaram diretamente na realização deste.

Os bons livros fazem “sacar” para fora o que a pessoa tem de melhor dentro dela.

Lina Sotis Francesco Moratti

AGRADECIMENTOS

À minha família, que sempre esteve presente.

Aos meus amigos, pelos empurrões e cobranças.

Ao meu orientador, por ter acreditado na conclusão deste trabalho.

RESUMO

A manutenção de software é a parte fundamental da engenharia de software que visa o acompanhamento das transformações e evolução de um sistema ao longo de seu ciclo de vida. A manutenção de software é dirigida por um processo, que por sua vez, controla todas as modificações que um software acarreta ao longo de sua existência. Um processo de manutenção bem definido tende a facilitar o trabalho dos mantenedores de software. Este trabalho tem a finalidade de apresentar uma ferramenta que auxilie no processo de manutenção de software. O modelo de manutenção escolhido pela ferramenta segue o padrão proposto pela ISO/IEC 14764.

Palavras chaves: Engenharia de Software; ISO/IEC 14764; Processo de Manutenção.

ABSTRACT

The software maintenance is the basic part of the software engineering that aims at the accompaniment of the transformations and evolution of a system throughout its cycle of life. The software maintenance is directed by a process, that in turn, controls all the modifications that a software causes throughout its existence. A clear-cut process of maintenance tends to facilitate the work of the mainteners of software. This work has the purpose to present a tool that assists in the process of software maintenance The model of maintenance chosen for the tool follows the standard considered for ISO/IEC 14764.

Keys-words: Engineering of Software; ISO/IEC 14764; Process of Maintenance.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

LISTA DE SIGLAS

CASE – *Computer Aided Software Engineering*

CSS – *Cascading Style Sheets*

HTML – *Hypertext Markup Language*

IEC – *Internacional Electrotechnical Commission*

IEEE – *Institute of Eletrical and Eletronic Engineers*

ISO – *Internacional Organization of Standardization*

PHP – *Hypertext Preprocessor*

UML – *Unified Modeling Language*

W3C – *World Wide Web Consortium*

WAE – *Web Application Extension*

SUMÁRIO

1 INTRODUÇÃO

Assim que um software é colocado em uso, novos requisitos emergem e os requisitos existentes são modificados à medida que a empresa que executa esse software passa por modificações. Partes do software podem precisar ser modificadas para corrigir os erros encontrados na operação, melhorar seu desempenho ou outras características não funcionais. Ou seja, depois de serem entregues, os sistemas de software sempre evoluem em resposta às exigências de mudanças. (SOMMERVILLE, 2003)

A manutenção de software, até muito recentemente, era a fase negligenciada do processo de engenharia de software. A literatura sobre manutenção contém muito poucos lançamentos quando comparada com as atividades de desenvolvimento. Relativamente pouca pesquisa ou dados de produção tem sido compilados sobre o assunto e poucas abordagens ou “métodos” técnicos têm sido propostos. (PRESSMAN, 1995)

Um dos problemas encontrados no processo de manutenção de software, segundo Sommerville (2003), refere-se à documentação. A documentação inexiste ou não representa o estágio atual do sistema. O reconhecimento de que o software deve ser documentado é um primeiro passo, mas a documentação deve ser compreensível e consistente com o código fonte para efetivamente auxiliar no processo. Freqüentemente é difícil ou impossível rastrear a evolução do software através de muitas versões ou lançamentos, devido às mudanças não estarem adequadamente documentadas.

Um outro problema refere-se ao crescente aumento dos custos da manutenção de software nas últimas décadas. Durante a década de 70, a manutenção era responsável por um índice de 40% do orçamento de software para uma organização de sistemas de informação. Já na década de 80, esse valor pulou para aproximadamente 60%. Se nada for feito para melhorar essa abordagem, muitas empresas gastarão aproximadamente 80% de seus orçamentos de softwares em manutenção nas próximas décadas, identificou Pressman (1995).

Segundo Scussiato (1998), fazer a manutenção de sistemas em tempo hábil e com custo e benefícios adequados é um desafio para muitos. Conhecer os sistemas antigos, os desenvolvimentos mais recentes e a automação dos processos são objetivos que muitas organizações tem como meta.

Sendo assim, este trabalho aborda a automatização de um processo de manutenção de software, com a implementação de uma ferramenta CASE (*Computer Aided Software Engineering*) para a *Web*, utilizando como base o modelo de manutenção proposto pela ISO/IEC 14764.

1.1 OBJETIVOS DO TRABALHO

Tendo em vista os problemas relatados, este trabalho tem como objetivo geral desenvolver uma ferramenta CASE para a *Web* direcionada ao processo de manutenção de software.

Os objetivos específicos são relatados a seguir:

a) a ferramenta deve manter a documentação dos sistemas, bem como registrar o histórico das alterações efetuadas;

b) o processo de manutenção adotado pela ferramenta CASE deve controlar a distribuição de tarefas entre membros de uma equipe de manutenção;

c) a CASE resultante deste projeto deve possuir interface para cliente, visando oferecer um relacionamento transparente entre cliente e empresa.

1.2 ESTRUTURA DO TRABALHO

A estrutura do trabalho foi definida seguindo os padrões propostos pela coordenadoria do curso. Sendo assim, o tópico seguinte refere-se a fundamentação teórica, onde são abordados temas relevantes como Manutenção de Software, Trabalhos Correlatos, Custos da Manutenção de Software, Modelos de Manutenção e Ferramentas CASE.

Outro tópico abordado refere-se ao desenvolvimento do trabalho. Para um melhor entendimento esta seção foi dividida em Requisitos Principais, Especificação, Implementação, Técnicas e Ferramentas Utilizadas, Operacionalidade da Implementação e Resultados. Vale ressaltar ainda, as Conclusões finais e as Extensões, este último com sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, são abordados alguns assuntos relevantes sobre o tema deste trabalho. Na primeira seção são relatados os conceitos, as definições e os tipos de manutenção de software. A segunda seção descreve os custos da manutenção de software. A terceira seção apresenta alguns modelos de manutenção existentes. A quarta seção contém algumas informações sobre ferramentas CASE e a última seção identifica alguns trabalhos correlatos e suas principais características

2.1 MANUTENÇÃO DE SOFTWARE

Segundo Sommerville (2003), a manutenção de software é o processo geral de modificação de um sistema depois que ele foi colocado em uso. As modificações podem ser simples, destinada a corrigir erros de código, mais extensas, a fim de corrigir os erros de projeto, ou significativas, com finalidade de corrigir erros de especificação ou acomodar novos requisitos.

Existem três diferentes tipos de manutenção de software identificados pela ISO/IEC 14764:

a) manutenção corretiva: destinada a reparar os defeitos de um software existente. De acordo com Sommerville (2003), a correção de erros de codificação é um processo relativamente barato; os erros de projeto são mais dispendiosos, uma vez que podem envolver a reprogramação de diversos componentes. Os erros de requisitos são os mais dispendiosos de corrigir, devido à extensiva atividade de re-projeto, que pode ser necessária;

b) manutenção adaptativa: são mudanças que não fazem parte da especificação do projeto e tem a finalidade de adaptar o software a um ambiente operacional diferente. Segundo Sommerville (2003), esse tipo de manutenção é necessário quando algum aspecto do ambiente de sistema é modificado, como o hardware, o sistema operacional da plataforma ou outras modificações no software de apoio. O sistema deve ser modificado, para que seja adaptado, a fim de lidar com essas mudanças ambientais;

c) manutenção perfectiva: faz acréscimo à funcionalidade do sistema. Esse tipo de manutenção, segundo Sommerville (2003), é necessária quando os requisitos de sistema são modificados, em resposta a mudanças organizacionais ou de negócios. A escala das mudanças requerida para o software é, freqüentemente, muito maior do que para outros tipos de manutenção.

As tarefas da manutenção adaptativa e perfectiva preocupam-se, principalmente, com o aprimoramento de um sistema de software. Segundo Peter e Pedrycz (2001), é evidente que as atividades de aprimoramento tendem a dominar a fase de manutenção dos sistemas de software. Com base em dados empíricos, a distribuição das tarefas em um esforço de manutenção fica em torno de 20% para a manutenção corretiva, 25% para a adaptativa e 55% para a perfectiva.

2.2 CUSTOS DA MANUTENÇÃO DE SOFTWARE

Os custos de manutenção do sistema representam uma grande proporção do orçamento da maioria das organizações que utilizam sistemas de software. Na década de 80, Lientz e Swanson constataram que as grandes organizações dedicavam, pelo menos, 50 por cento do total de seu esforço de programação para a evolução dos sistemas já existentes. McKee (1984 apud SOMMERVILLE, 2003) constatou uma distribuição similar de esforço de manutenção em diferentes tipos de manutenção, mas sugeriu que a quantidade de esforço gasto em manutenção é de cerca de 65 a 75 por cento do esforço total disponível. Uma vez que as organizações substituíram os antigos sistemas por sistemas de prateleira, como os sistemas de planejamento de recursos corporativos, esse número pode não ter diminuído. Embora os detalhes possam ser incertos, sabe-se que as alterações em software continuam sendo um custo importante para todas as organizações. (SOMMERVILLE, 2003)

2.3 MODELOS DE MANUTENÇÃO

Vários modelos de manutenção foram desenvolvidos desde a década de 1970. Os primeiros modelos tinham a manutenção corretiva como ponto principal. Um exemplo de modelo corretivo é fornecido por Sharpley. O modelo de Sharpley concentrava-se na verificação de problemas, no diagnóstico de problemas, na reprogramação e em verificar se o código corrigido satisfazia os requisitos do documento *baseline*. Os modelos de processo preponderaram no pensamento de manutenção de software recente. Três exemplos são os

modelos de manutenção de software propostos por Taute, pela IEEE e pela ISO. (PETER E PEDRYCZ, 2001)

2.3.1 Modelo de Manutenção de Taute

De acordo com Peter e Pedrycz (2001), o modelo de Taute foi introduzido por Taute e elaborado por Parikh. Trata-se de um modelo direto, prático e de fácil compreensão. O ciclo inicia-se com uma solicitação de mudança e termina com a operação bem-sucedida de um produto de software modificado. O modelo de Taute possui oito fases:

a) Solicitação: contém as informações necessárias para processar uma solicitação de mudança;

b) Estimativa: o tempo, custos, recursos e o impacto da solicitação de mudança são determinados;

c) Agendamento: é determinado o lançamento da próxima versão de um produto de software, nesta fase os documentos de planejamento são preparados;

d) Programação: uma cópia da versão de produção do software a ser modificado é liberada, e começa a criação de uma versão de teste;

e) Testes: uma cópia recém modificada do software de produção é testada;

f) Documentação: após a conclusão bem sucedida dos testes de um novo produto, a documentação existente é atualizada;

g) Lançamento: o novo sistema e sua documentação atualizada são disponibilizados e dá-se inicio aos testes de aceitação do software;

h) Operacional: a disponibilização e a operação da nova versão começam após a conclusão bem sucedida dos testes de aceitação.

Peter e Pedrycz (2001) identificam que esse modelo destaca-se dos demais por enfatizar as versões programadas dos produtos de software resultantes das solicitações de mudanças. A figura 01 apresenta o funcionamento do modelo de manutenção de Taute:

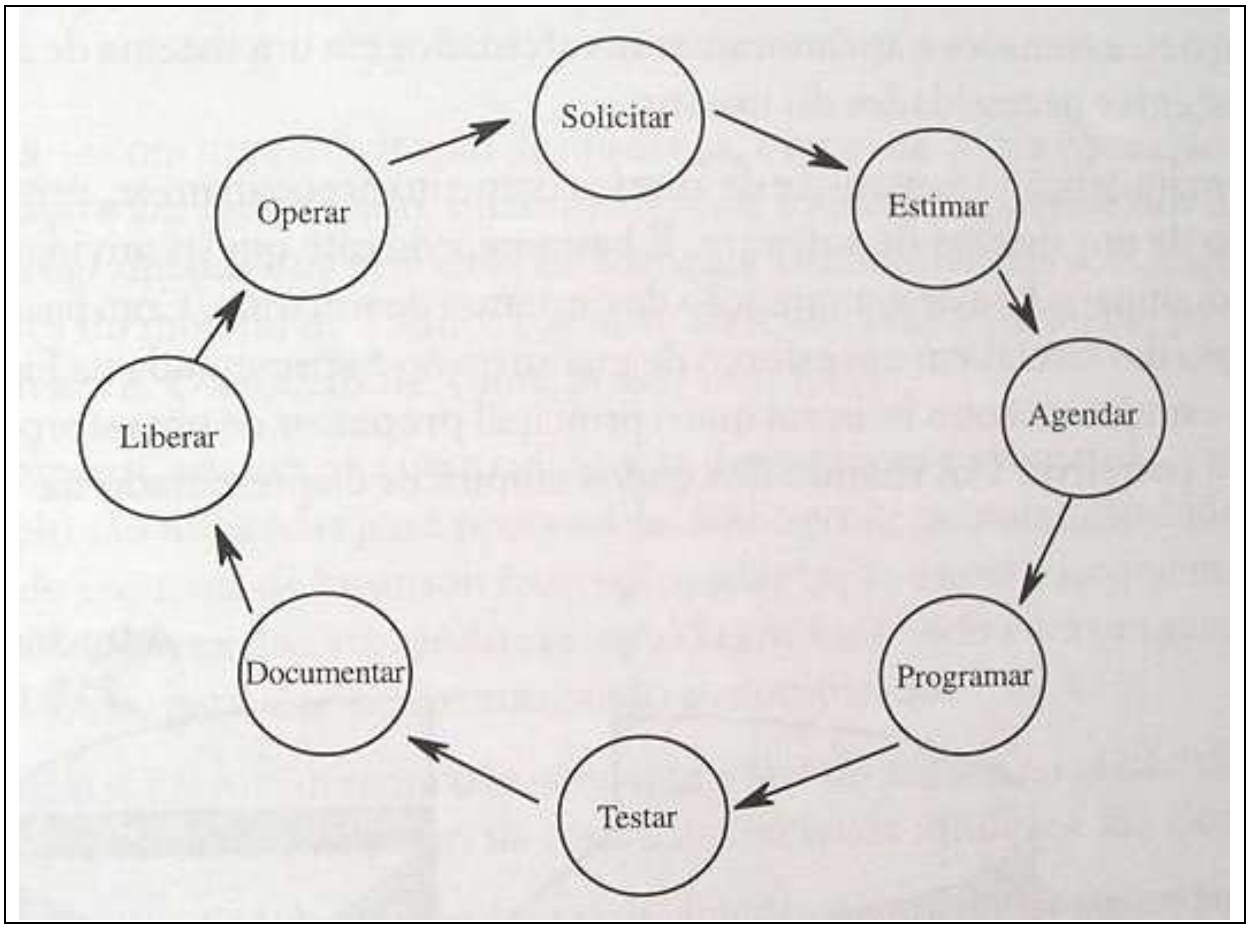


FIGURA 01 – Modelo de manutenção de Taute

2.3.2 Modelo de Manutenção segundo a IEEE 1219

Segundo Peter e Pedrycz (2001), o modelo é disparado por uma solicitação de mudança. O padrão associa os mecanismos de entrada, saída e controle relativos a cada fase do processo de manutenção. Esses mecanismos são detalhados a seguir:

a) entrada: documentos de contrato do projeto, saída da fase de análise, código-fonte, banco de dados;

b) saída: lista de modificações revisada, *baseline* de projeto atualizada, planos de teste atualizados, análise detalhada revisada, requisitos verificados, plano de implementação revisado, restrições documentadas e riscos documentados;

c) controle: inspecionar software, verificar documentação de projeto, rastrear projeto de acordo com os requisitos.

Peter e Pedrycz (2001) identificam que o modelo de manutenção da IEEE é aplicável não somente a fase de manutenção, como também, a qualquer fase de um ciclo de vida típico de software. A figura 02 apresenta o funcionamento do modelo de manutenção da IEEE:

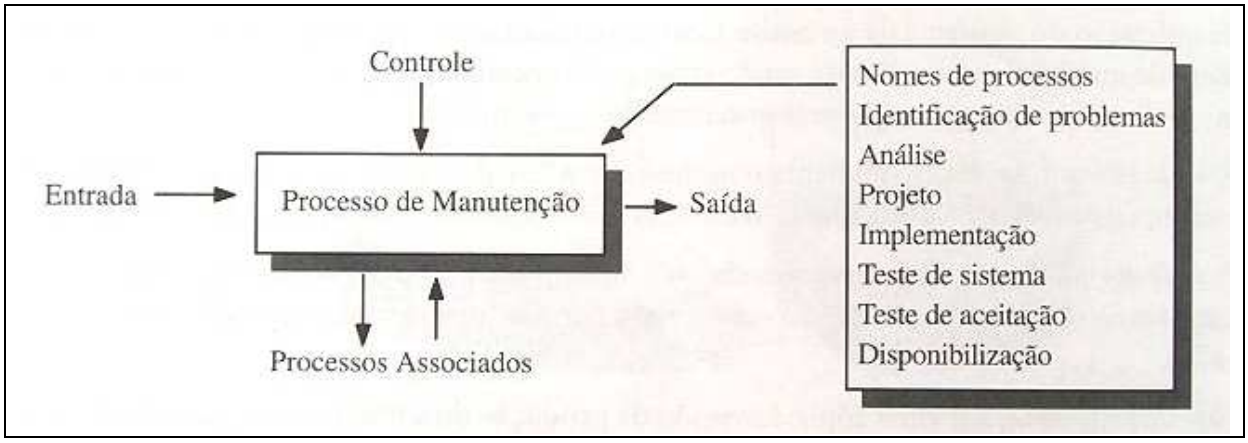


FIGURA 02 – Modelo de manutenção da IEEE 1219

2.3.3 Modelo de Manutenção segundo a ISO/IEC 14764

O Processo de Manutenção (PM) contém as atividades e tarefas necessárias para modificar um software existente com o intuito de preservar sua integridade. Essas atividades e tarefas são de responsabilidade da equipe de manutenção (EM). Este padrão internacional fornece exemplos e etapas a serem seguidas para implementar as atividades e tarefas da manutenção. A EM deve assegurar-se de que o PM seja consistente e funcional antes da liberação do software. O PM será ativado após a solicitação de mudanças no software existente.

As atividades que compreendem o PM, segundo a ISO/IEC 14764, são:

a) Execução do processo;
b) Problema e análise da manutenção;
c) Implementação da manutenção;
d) Aceitação e revisão da manutenção;
e) Migração;
f) Encerramento.

Cada atividade do PM possui entradas, tarefas, controles, suporte e saídas. As entradas são transformadas ou consumidas pela atividade da manutenção para produzir saídas. As tarefas são instruções a serem seguidas que auxiliam na execução da atividade da

manutenção. Os controles fornecem a orientação para assegurar-se de que a atividade da manutenção produza saídas corretas. O suporte contempla os recursos ou objetos utilizados para consulta. As saídas são os dados ou os objetos produzidos pela atividade da manutenção. A figura 03 apresenta o modelo de manutenção da ISO. Os nomes das atividades foram traduzidos para um melhor entendimento. Os próximos tópicos detalham o funcionamento de cada atividade proposta pela ISO/IEC 14764:

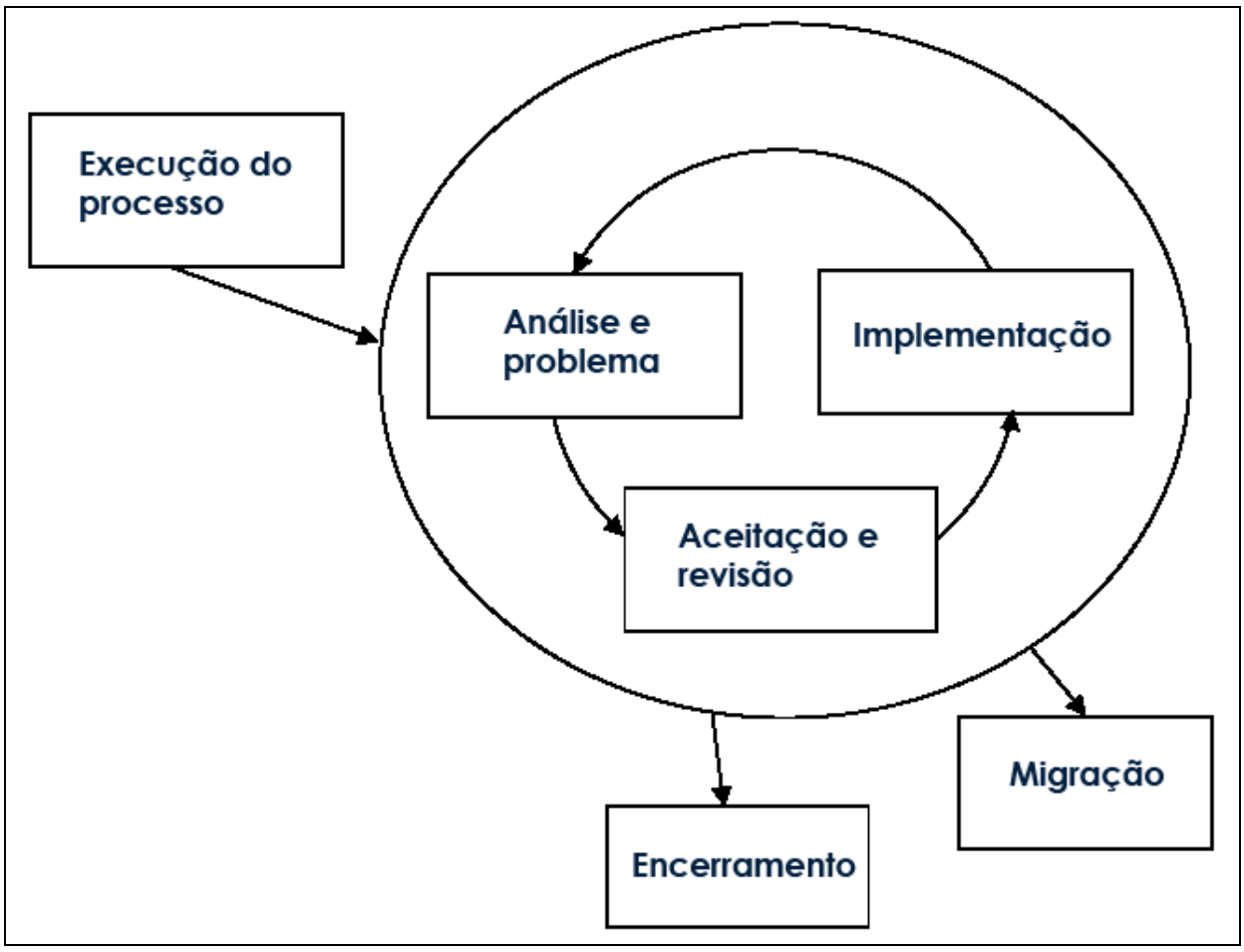


FIGURA 03 – Modelo de manutenção da ISO/IEC 14764

2.3.3.1 EXECUÇÃO DO PROCESSO

Durante a execução do processo, a EM deve estabelecer a estratégia e os procedimentos que deverão ser empregados na execução do PM. As entradas desta atividade incluem a *baseline*, a documentação do sistema e uma solicitação de mudança (SM) ou um relatório de problema (RP).

A fim de desenvolver uma estratégia e procedimentos adequados, a EM deve identificar os limites da manutenção, buscar alternativas para a análise da manutenção, designar alguém para realizar a manutenção, verificar os recursos disponíveis, estimar os custos da manutenção, avaliar a manutenibilidade do sistema, determinar as exigências da transição, verificar o histórico da transição e identificar o PM a ser usado.

Para implementar os procedimentos na SM/RP corretamente a EM deve criar um identificador, criar categorias e prioridades, criar procedimentos para determinar as análises de tendência, determinar procedimentos para que um operador submeta uma SM/RP, determinar como o *feedback* inicial será fornecido aos usuários, determinar como o ciclo de trabalho será fornecido aos usuários, determinar como os dados serão incorporados à base de dados de acordo com o *status* atual da manutenção e determinar a próxima seqüência do *feedback* fornecida aos usuários.

As saídas desta atividade são: a estratégia da manutenção, os procedimentos da manutenção, procedimentos para definição do problema, *feedback* para os usuários e a gerência de configuração.

2.3.3.2 ANÁLISE E PROBLEMA DA MANUTENÇÃO

Durante a atividade de análise e problema da manutenção, a EM deve analisar a SM/RP, entender o problema, desenvolver opções para implementar a manutenção, documentar o SM/RP com os resultados e as opções de implementação, obter a aprovação da opção desenvolvida. As entradas desta atividade incluem a SM/RP, *baseline* atualizada, repositório do software, documentação do sistema (requisitos funcionais, requisitos não-funcionais) e as saídas da atividade de execução do processo.

Antes de modificar o sistema, a EM deve analisar a SM/RP para determinar seu impacto na organização, no sistema existente e nos sistemas de integração. As tarefas correspondentes são: desenvolver e documentar soluções em potencial e obter a aprovação para executar a solução desejada.

Durante a análise da SM/RP, deve-se levar em consideração o tipo da manutenção, os limites da manutenção (recursos disponíveis, custos e tempo) e o desempenho da manutenção (performance, segurança e integridade). Após a aprovação da solução escolhida, deve-se definir ou atualizar a prioridade da manutenção.

As saídas desta atividade são: análise do impacto (indicação do problema ou nova exigência, tipo de manutenção, estimativas de custo e tempo e prioridade), solução recomendada, aprovação da manutenção, documentação atualizada.

2.3.3.3 IMPLEMENTAÇÃO DA MANUTENÇÃO

Durante a atividade de implementação da manutenção, a EM deve realizar o processo de desenvolvimento que corresponde à codificação e teste das modificações no software existente. As entradas desta atividade incluem a *Baseline* atualizada, SM/RP aprovada e as saídas da atividade de problema e análise da manutenção.

Antes que o processo de desenvolvimento seja iniciado, a EM deve identificar os elementos do software que necessitem de atualização, identificar os elementos de integração afetados e identificar a documentação a ser alterada.

As saídas desta atividade são: documentação atualizada (registros modificados, análise detalhada, requisitos atualizados), os fontes modificados e o resultado dos testes.

2.3.3.4 ACEITAÇÃO E REVISÃO DA MANUTENÇÃO

Durante a atividade de aceitação e revisão da manutenção, a EM deve assegurar-se de que as modificações no sistema estão corretas e que foram realizadas de acordo com os padrões estabelecidos. As entradas desta atividade incluem o software modificado, o resultado dos testes sobre o software modificado e as saídas da atividade de implementação da manutenção.

As revisões são realizadas a fim de obter a aprovação e conclusão satisfatória das modificações. Durante a revisão a EM deve rastrear as exigências da SM/RP desde o projeto ao código-fonte, verificar a funcionalidade do código, verificar se o código está de acordo com os padrões da empresa, verificar os componentes de software utilizados, verificar se os componentes novos foram integrados de forma apropriada, checar se a documentação foi atualizada e realizar os testes no software modificado.

As saídas desta atividade são: nova *baseline* com as modificações aceitas, as modificações rejeitadas, um aval de aceitação da manutenção e o resultado dos testes realizados.

2.3.3.5 MIGRAÇÃO

Durante a vida útil de um sistema, o mesmo pode sofrer transformações para funcionar em ambientes diferentes. A fim de migrar o sistema para um ambiente novo, a EM deve realizar um levantamento das etapas necessárias para posteriormente efetuar a migração. As entradas desta atividade incluem o ambiente antigo, o ambiente novo, a *baseline* antiga e a *baseline* nova.

Antes de efetuar a migração, a EM deve identificar todos os produtos ou dados do software afetados, desenvolver um plano para a migração, notificar os usuários da migração, fornecer treinamento, fornecer uma notificação de conclusão, avaliar o impacto do ambiente novo e manter *backup* dos produtos e dados antigos.

O plano para a migração contém as etapas propostas pela ISO/IEC 12207, na qual a EM deve analisar os requisitos da migração, determinar o impacto da migração no software, estabelecer uma programação para execução da migração, identificar as ferramentas de migração necessárias, desenvolver as ferramentas de migração, priorizar a conversão dos produtos e dados do software, realizar a conversão dos produtos e dados do software, executar a migração no novo ambiente, verificar o resultado da migração e manter suporte ao ambiente antigo.

As saídas desta atividade são: o plano de migração, ferramentas de migração, o software migrado, notificação de conclusão e *backup* dos produtos e dados antigos.

2.3.3.6 ENCERRAMENTO

Quando um software alcança o final de sua vida útil, deve ser aposentado. Uma análise deve ser realizada a fim de verificar se o software ainda tem condições de competir no mercado. Caso o software não tenha condições de continuar no mercado, a EM deve determinar as ações necessárias para realizar sua retirada. As entradas desta atividade incluem o software antigo, o software novo, o ambiente antigo e o ambiente novo.

Antes de efetuar a retirada do software, a EM compromete-se em desenvolver um plano para a retirada do software, notificar os usuários, fornecer uma notificação de conclusão e manter *backup* dos produtos e dados antigos.

O plano para a retirada do software contém as etapas propostas pela ISO/IEC 12207, na qual a EM deve analisar os requisitos para a retenção, determinar o impacto do software retido, estabelecer uma programação para execução da retenção e manter a responsabilidade de suporte futuro.

As saídas desta atividade são: o plano de retirada do software, o software retido, notificação de conclusão e *backup* do software antigo.

2.4 FERRAMENTAS CASE

A engenharia de software apoiada por computador (*Computer Aided Software Engineering*, CASE) pode ser uma simples ferramenta que apóia uma atividade específica de engenharia de software ou uma complexa ferramenta, como um “ambiente” completo que inclui uma base de dados, pessoal, *hardware*, uma rede, sistemas operacionais, normas e miríades de outros componentes. (PRESSMAN, 2002)

CASE são ferramentas que reduzem a quantidade de esforço necessário para produzir um trabalho ou atingir algum marco de projeto com benefícios substanciais. Além disso, de acordo com Pressman (2002), podem oferecer novos modos de olhar a informação de engenharia de software, modos que aperfeiçoam o conhecimento do engenheiro que está realizando o trabalho. Isso conduz a decisões mais elaboradas e a qualidade superior de software.

2.4.1 Taxonomia de Ferramentas CASE

Segundo Pressman (2002), as ferramentas CASE podem ser classificadas por função, por seu papel como instrumentos para gerentes ou pessoal técnico, por seu uso nos vários passos do processo de engenharia de software, pela arquitetura do ambiente (hardware e software) que as apóia, ou mesmo por sua origem ou custo.

A Taxonomia apresentada neste trabalho usa função como critério principal. Fazem parte da taxonomia CASE as ferramentas de engenharia de processo de negócio, modelagem e gestão de processo, planejamento de projeto, análise de riscos, gestão de projetos, rastreamento de requisitos, métricas e gestão, documentação, software básico, garantia de qualidade, gestão de base de dados, gestão de configuração de software, análise e projeto, projeto e desenvolvimento de interfaces, prototipação, programação, desenvolvimento da

Web, integração e teste, análise estática, análise dinâmica, gestão de testes, teste cliente/servidor e as ferramentas de reengenharia.

De acordo com a taxonomia citada, a CASE resultante deste trabalho enquadra-se nas ferramentas de modelagem e gestão de processo, que segundo Pressman (2002), são ferramentas usadas para representar os elementos principais de um processo, de modo a entendê-lo melhor. Tais ferramentas também podem fornecer ligações para descrições de processo, que ajudam os envolvidos no processo a entender as tarefas de trabalho necessárias para realizá-lo.

2.5 TRABALHOS CORRELATOS

Esta seção apresenta alguns trabalhos correlatos, identificando suas principais características e autores:

a) “Software de apoio à manutenção de sistemas baseado em normas de qualidade”, Hoppe (1999): refere-se a um sistema intranet para uso em rede local da empresa. Os processos são definidos de acordo com as normas de qualidade ISO/IEC 12207, ISO/IEC 9000-3 e ISO/IEC 9000-2. Na implementação e codificação do sistema foi utilizado o *Delphi 3.0* como ambiente de desenvolvimento e o *Paradox* para armazenamento dos dados;

b) “Processo de manutenção de sistemas baseado na norma ISO/IEC 12207”, Scussiato (1998): o processo de manutenção é definido de acordo com a norma ISO/IEC 12207. A norma ISO/IEC 12207 abrange os processos do ciclo de vida do software e tem como principal objetivo fornecer estrutura e linguagem comuns para todos os envolvidos com o desenvolvimento do software.

3 DESENVOLVIMENTO DO TRABALHO

Esta seção aborda a metodologia de desenvolvimento do trabalho. O tópico inicial descreve os requisitos atendidos pela ferramenta. O próximo tópico refere-se a especificação da ferramenta, onde são apresentados alguns diagramas UML (*Unified Modeling Language*) visando uma melhor compreensão da ferramenta. O tópico seguinte refere-se a implementação da ferramenta, onde está descrito seu funcionamento, tecnologias utilizadas, operacionalidade e resultados obtidos.

3.1 REQUISITOS

O quadro 1 lista os requisitos funcionais atendidos pela ferramenta:

Requisitos funcionais
RF 1. Sistema armazena histórico para cada manutenção
RF 2. Sistema identifica o estágio atual para cada manutenção
RF 3. Todos os usuários efetuam o login no sistema
RF 4. Todos os usuários consultam manutenções
RF 5. Solicitante faz a solicitação de manutenção via WEB
RF 6. Solicitante aprova orçamento da manutenção
RF 7. Solicitante homologa manutenção
RF 8. Coordenador define prioridades para a manutenção
RF 9. Coordenador identifica tipo de manutenção
RF 10. Coordenador define um Analista para a manutenção
RF 11. Analista define a estimativa de custo para a manutenção
RF 12. Analista define a estimativa de tempo para a manutenção
RF 13. Analista anexa os documentos modificados na manutenção
RF 14. Analista define um programador para a manutenção
RF 15. Analista aprova a lista de produtos modificados pela manutenção
RF 16. Programador anexa os fontes modificados na manutenção
RF 17. Programador anexa um instalador do software para o Solicitante
RF 18. Coordenador ou Analista mantém fontes
RF 19. Coordenador ou Analista mantém sistemas
RF 20. Coordenador ou Analista mantém versões
RF 21. Coordenador ou Analista mantém módulos

QUADRO 1 – Requisitos funcionais

O quadro 2 lista os requisitos não funcionais atendidos pela ferramenta:

Requisitos não funcionais
RNF 1. Sistema Operacional Windows
RNF 2. Banco de Dados *Firebird 1.5*
RNF 3. Linguagem PHP (*Hypertext Preprocessor*) 5 ou superior
RNF 4. Servidor IIS (*Internet Information Services*)
RNF 5. Suporte a *Javascript* e CSS (*Cascading Style Sheets*)

QUADRO 2 – Requisitos não funcionais

3.2 ESPECIFICAÇÃO

Esta seção descreve os modelos e diagramas desenvolvidos durante o trabalho. Os primeiros tópicos tratam, respectivamente, os diagramas UML de Atividades e Caso de Uso. Ambos foram desenvolvidos utilizando o *Enterprise Architect 3.51*. O último tópico apresenta o modelo ER (Entidade e Relacionamento) gerado a partir do *Power Designer 9.0*.

3.2.1 DIAGRAMA DE ATIVIDADES

A figura 04 representa o Diagrama de Atividades. Trata-se de um *workflow* definido sob o processo de manutenção adotado e identificado na seção 2.3.3. O processo é disparado por uma solicitação de manutenção, que então é avaliada, analisada, aprovada, implementada, testada e finalmente homologada.

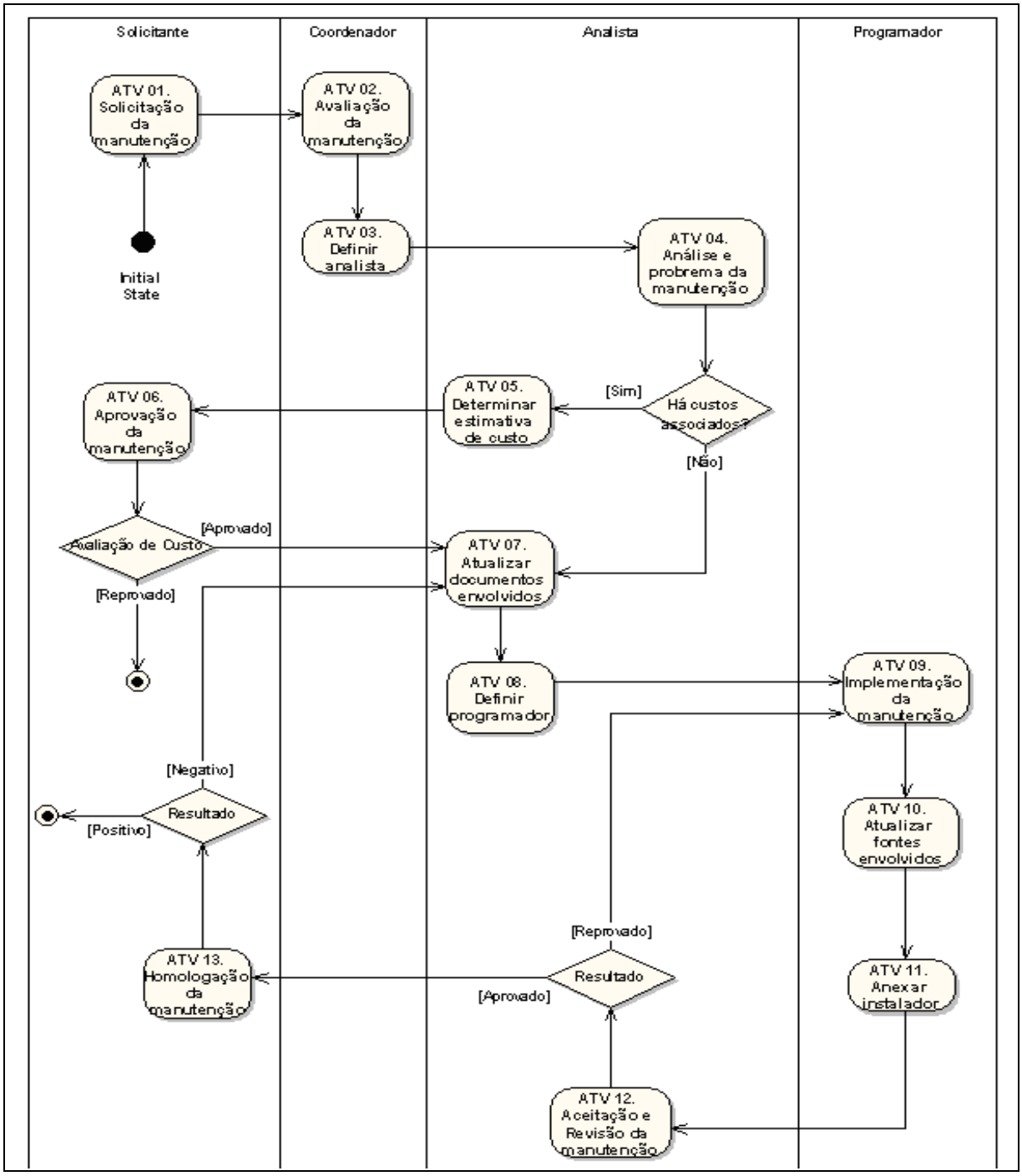


FIGURA 04 – Diagrama de Atividades

3.2.2 DIAGRAMA DE CASO DE USO

A figura 05 representa o Diagrama de Pacotes. Este diagrama faz referência aos requisitos funcionais do sistema. Cada pacote representa as funcionalidades exclusivas de um ator, exceto o pacote geral que contém as funcionalidades comuns realizadas por mais de um ator. A descrição dos cenários encontra-se na seção Apêndice A.

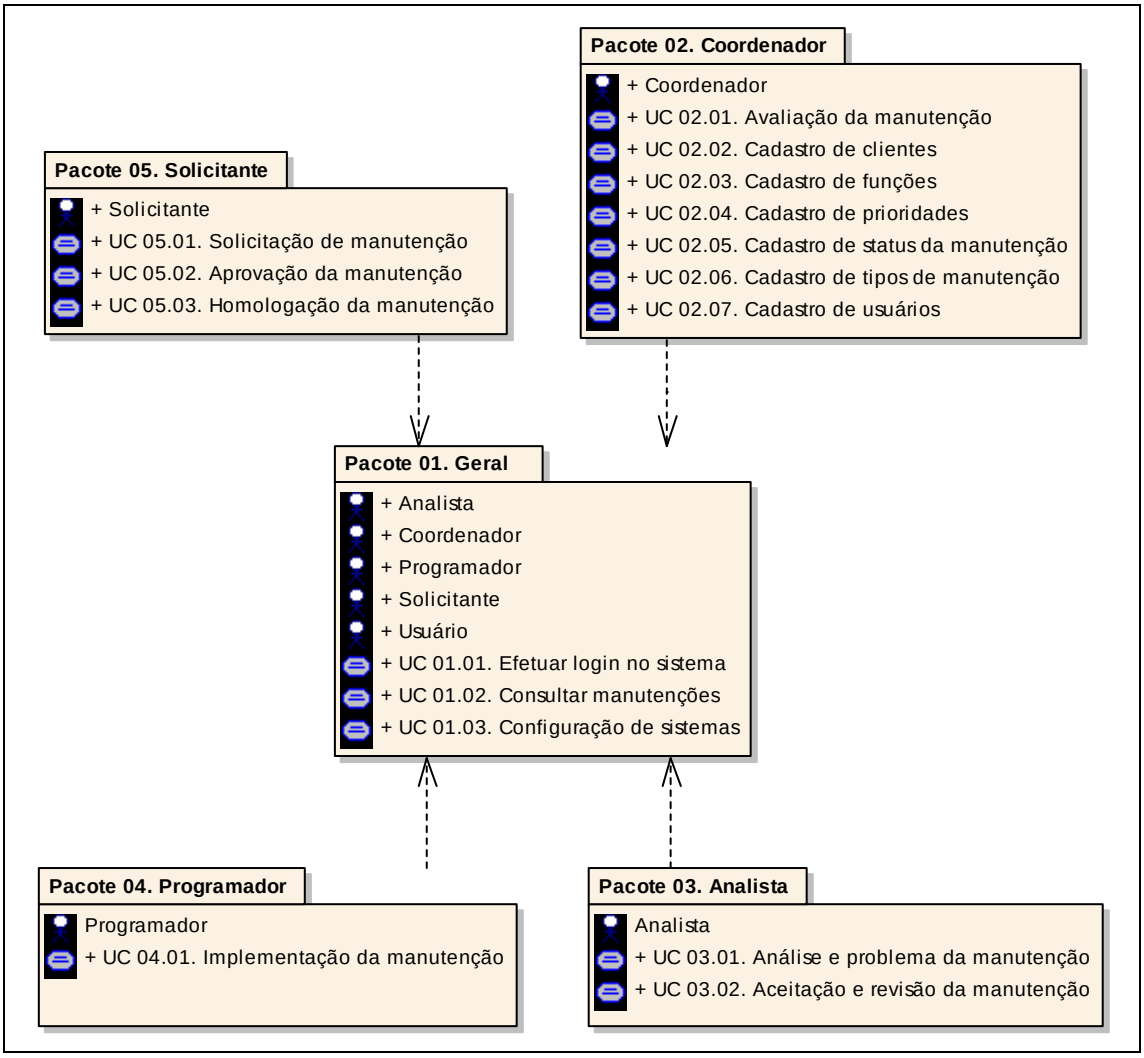


FIGURA 05 – Diagrama de Pacotes

As próximas ilustrações identificam as funcionalidades existentes em cada pacote. A figura 06 representa as funcionalidades gerais dos atores. A figura 07 detalha as funcionalidades exclusivas do Coordenador. A figura 08 identifica as funcionalidades exclusivas do Analista. A figura 09 mostra as funcionalidades exclusivas do Programador e a figura 10 apresenta as funcionalidades exclusivas do Solicitante.

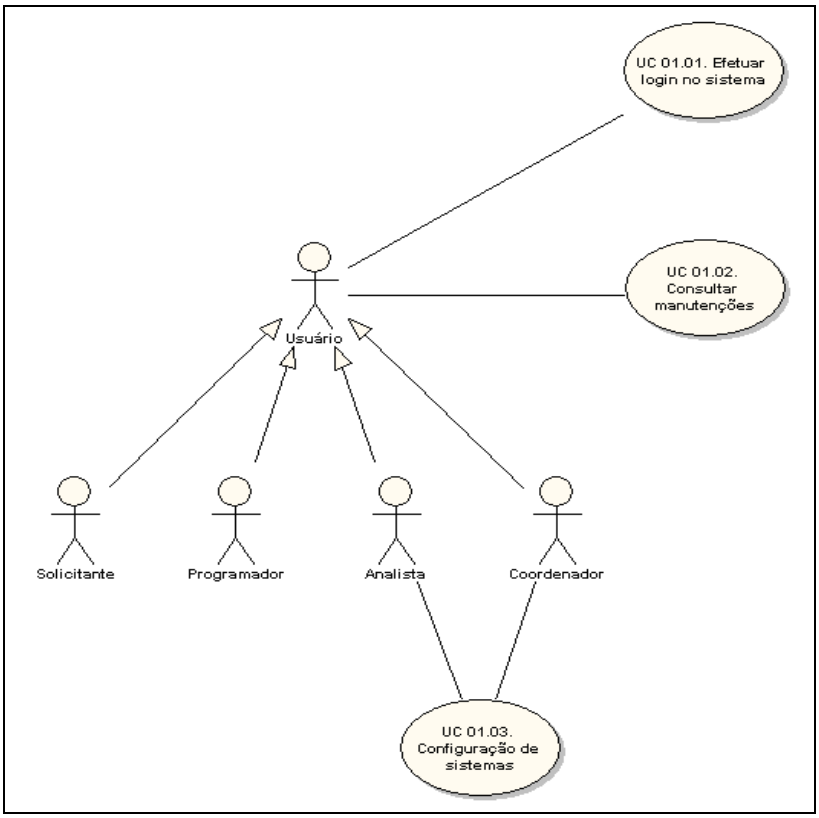


FIGURA 06 – Pacote 01. Geral, identifica as funcionalidades comuns dos atores

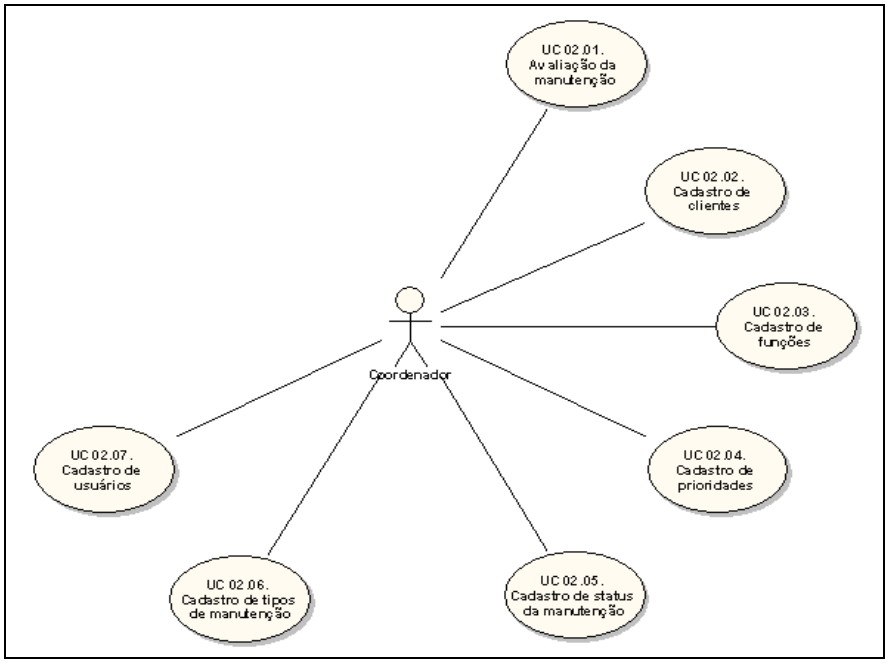


FIGURA 07 – Pacote 02. Coordenador, identifica as funcionalidades do Coordenador

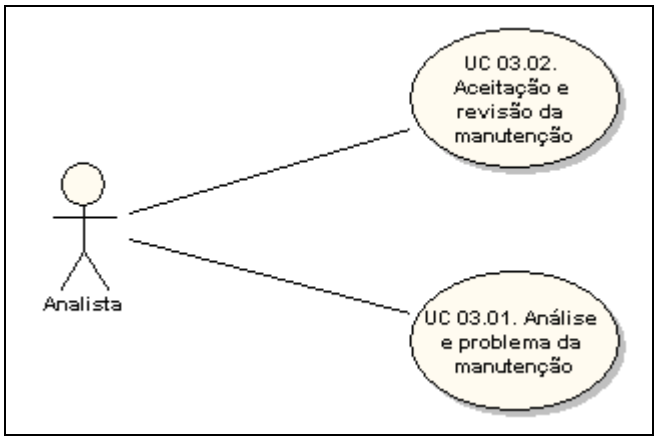


FIGURA 08 – Pacote 03. Analista, identifica as funcionalidades do Analista

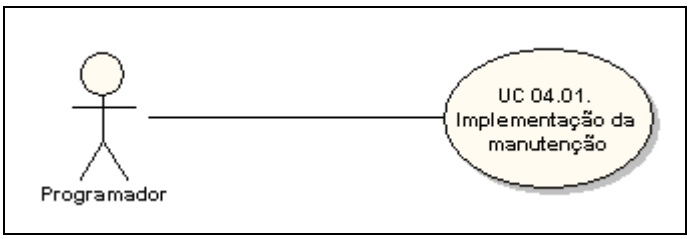


FIGURA 09 – Pacote 04. Programador, identifica as funcionalidades do Programador

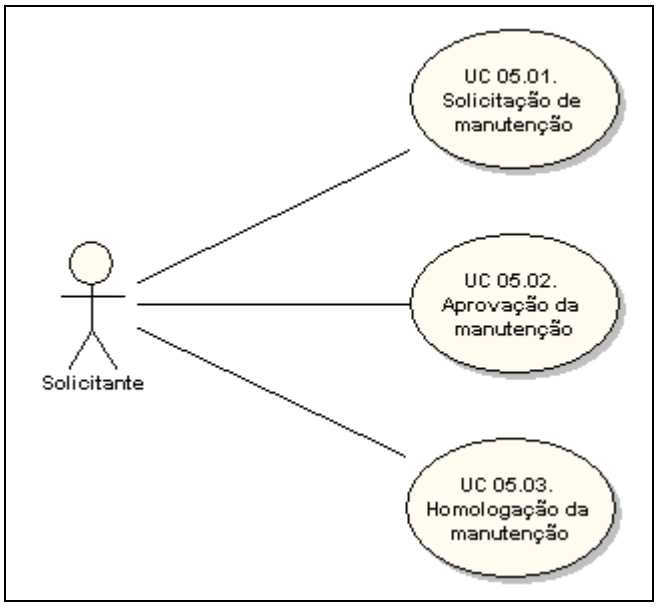


FIGURA 10 – Pacote 05. Solicitante, identifica as funcionalidades do Solicitante

3.2.3 DIAGRAMA DE CLASSES

A figura 11 apresenta as principais classes da ferramenta desenvolvida, objetivando uma visão do domínio do problema.

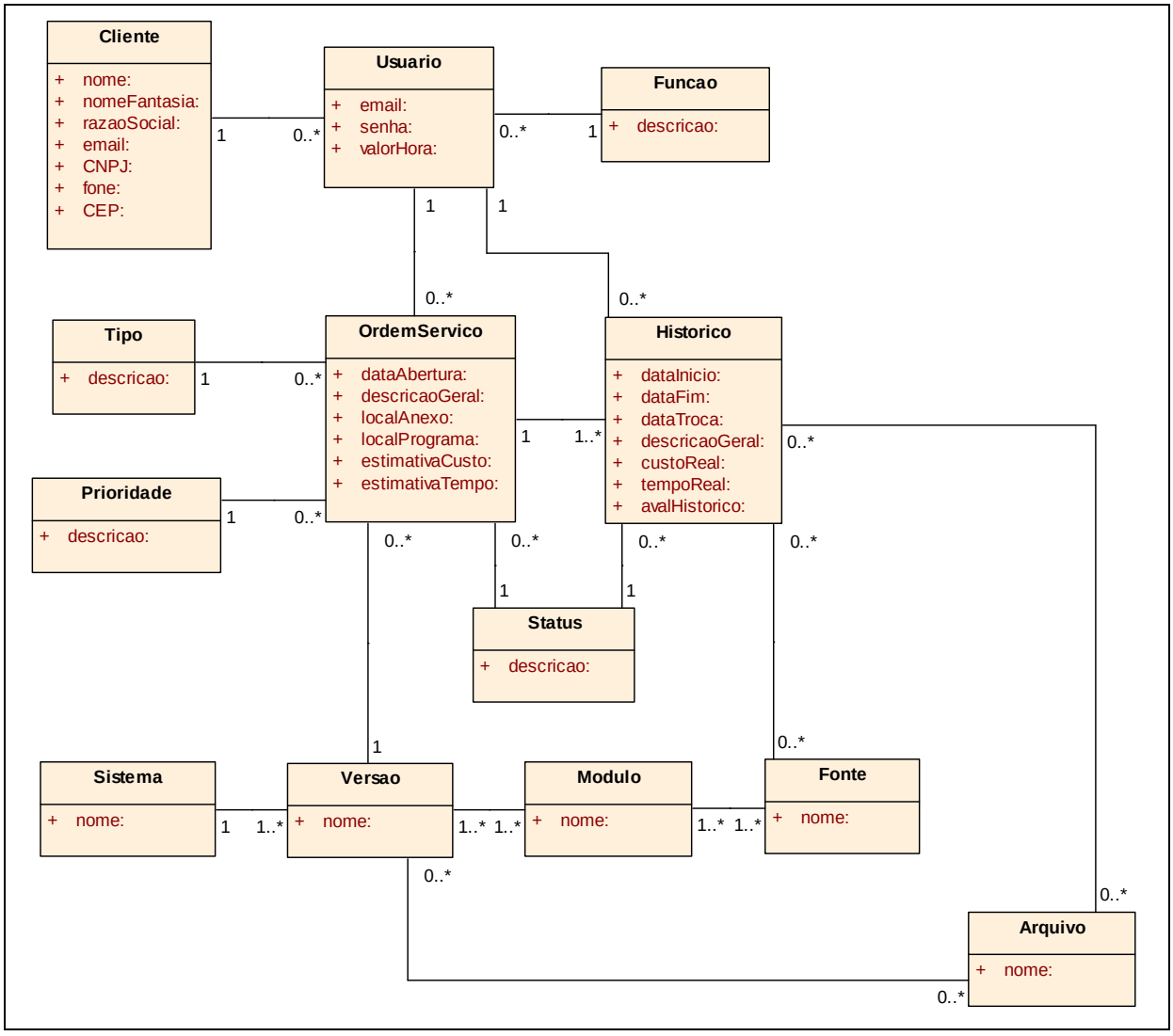


FIGURA 11 – Diagrama de classes de domínio

3.2.4 MODELO DE CLASSES WAE

Segundo Conallen (2003), o modelo de classes WAE (*Web Application Extension*), permite representar páginas *web* e outros elementos significativos do ponto de vista arquitetônico. O modelo representado na figura 12 detalha o funcionamento da tela de manutenções pendentes, *interface* principal da ferramenta. Este modelo está vinculado ao *workflow* apresentado na seção 3.2.1, ou seja, refere-se apenas as atividades principais não contemplando as funcionalidades de suporte ao *workflow*, tais como cadastro de funções, usuários, dentre outras.

De acordo com a figura 12, observa-se que o modelo é disparado pela solicitação de uma *server-page*. O servidor interpreta a *server-page* e monta a *client-page* com as pendências do usuário logado. A *client-page* contém os *links* necessários de acesso aos pacotes, que representam as telas para edição da manutenção pendente.

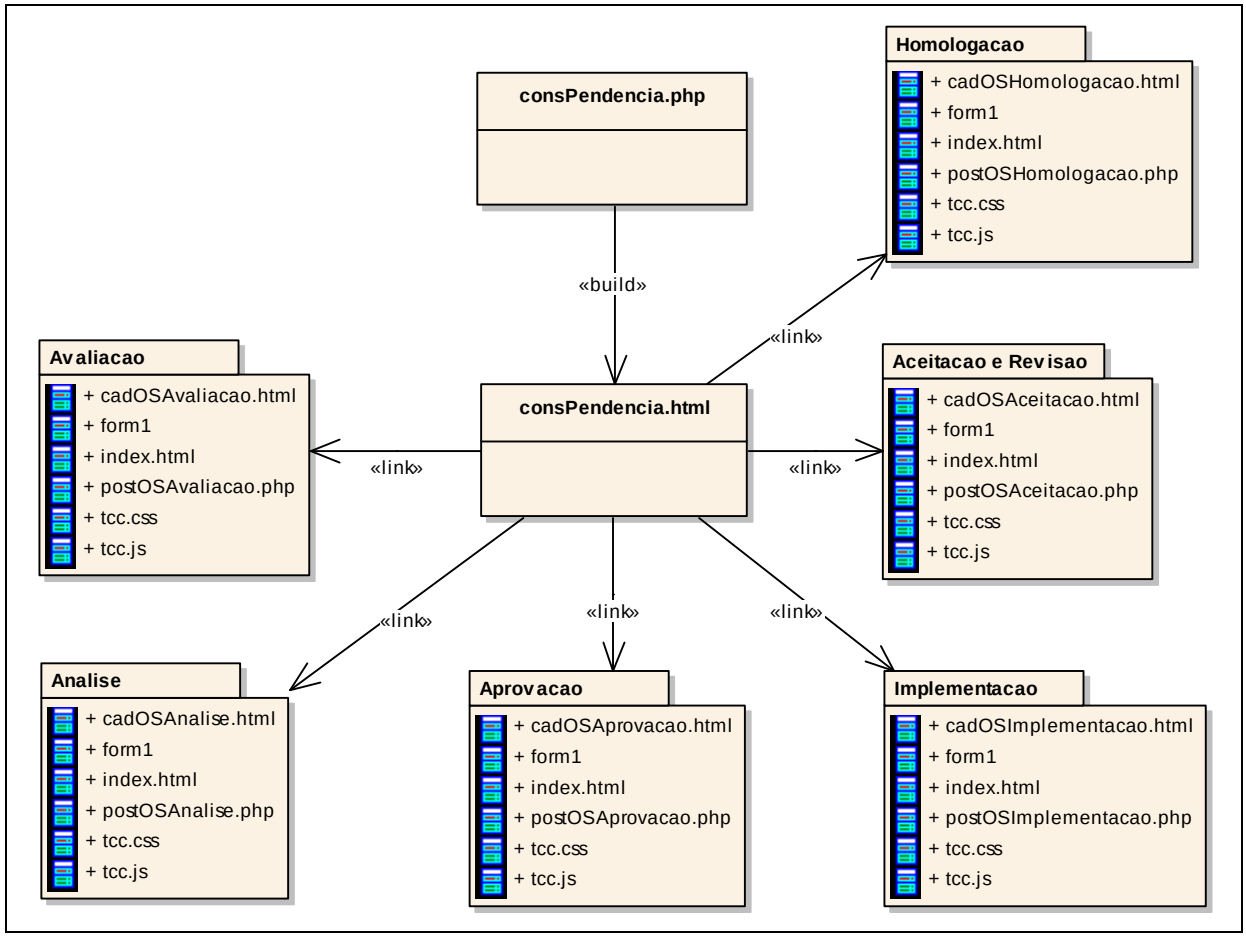


FIGURA 12 – Representação WAE das manutenções pendentes

As figuras seguintes detalham o funcionamento dos pacotes identificados na figura 12. Todos os pacotes seguem o mesmo padrão de funcionamento, proposto pela figura 13. O pacote é disparado por uma *client-page* que faz uso das funcionalidades de duas classes com estereótipos distintos. A primeira classe é uma *client-script*, que contém código em *Javascript* utilizado principalmente para navegação e validação do formulário. A outra classe é uma *style-sheet*, que contém código em CSS cuja finalidade é manter o padrão visual da página. Observa-se também, que a classe com estereótipo *form*, aponta para uma *server-page* que tem como finalidade armazenar os dados do formulário no banco de dados. Após o processamento da *server-page* a mesma faz o redirecionamento para uma *client-page*. A figura 13 detalha o funcionamento do pacote avaliação. Observa-se que o form1 contém o campo cmd utilizado para concluir ou não a etapa da manutenção, este atributo está presente em diversos *forms*, e refere-se ao controle do *workflow* – indicando etapa concluída.

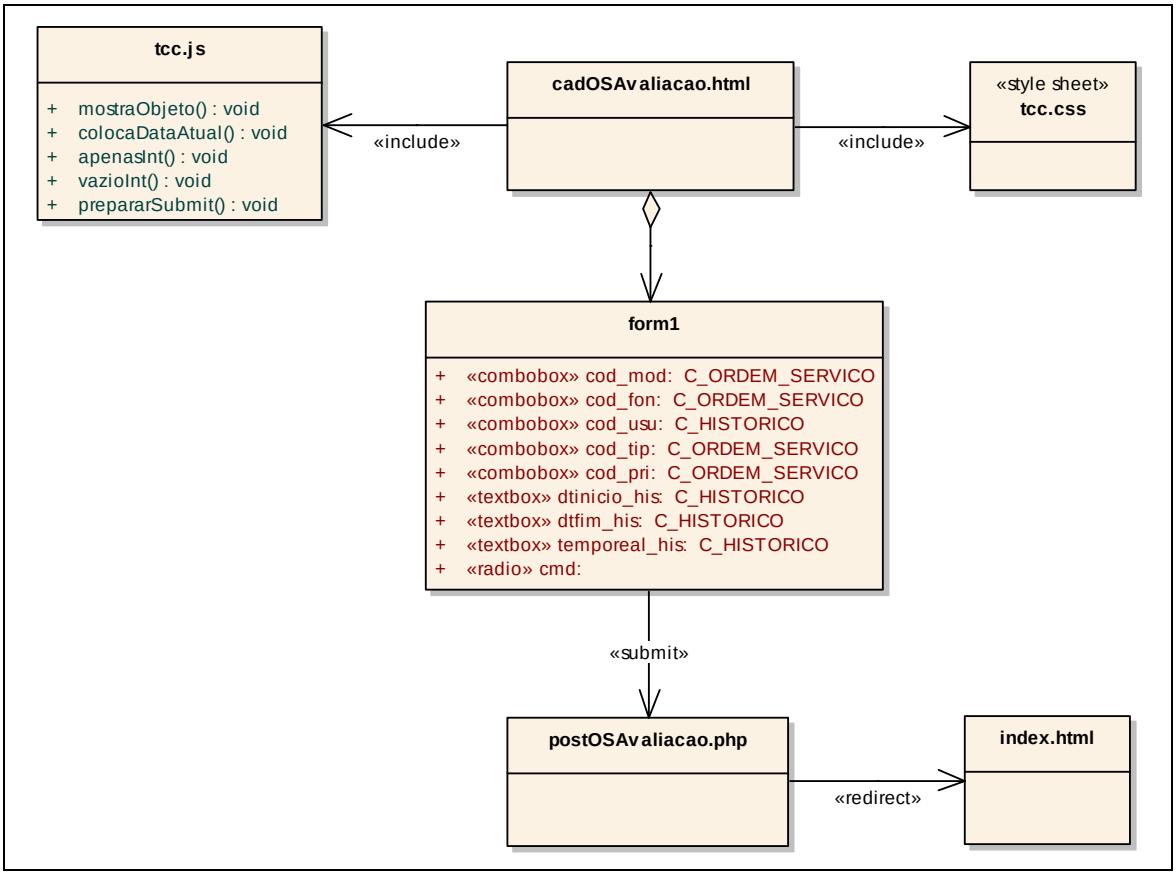


FIGURA 13 – Representação WAE do pacote avaliação

A figura 14 detalha o funcionamento do pacote análise. Observa-se que o form1 contém o campo env utilizado para encaminhar a manutenção para uma etapa específica.

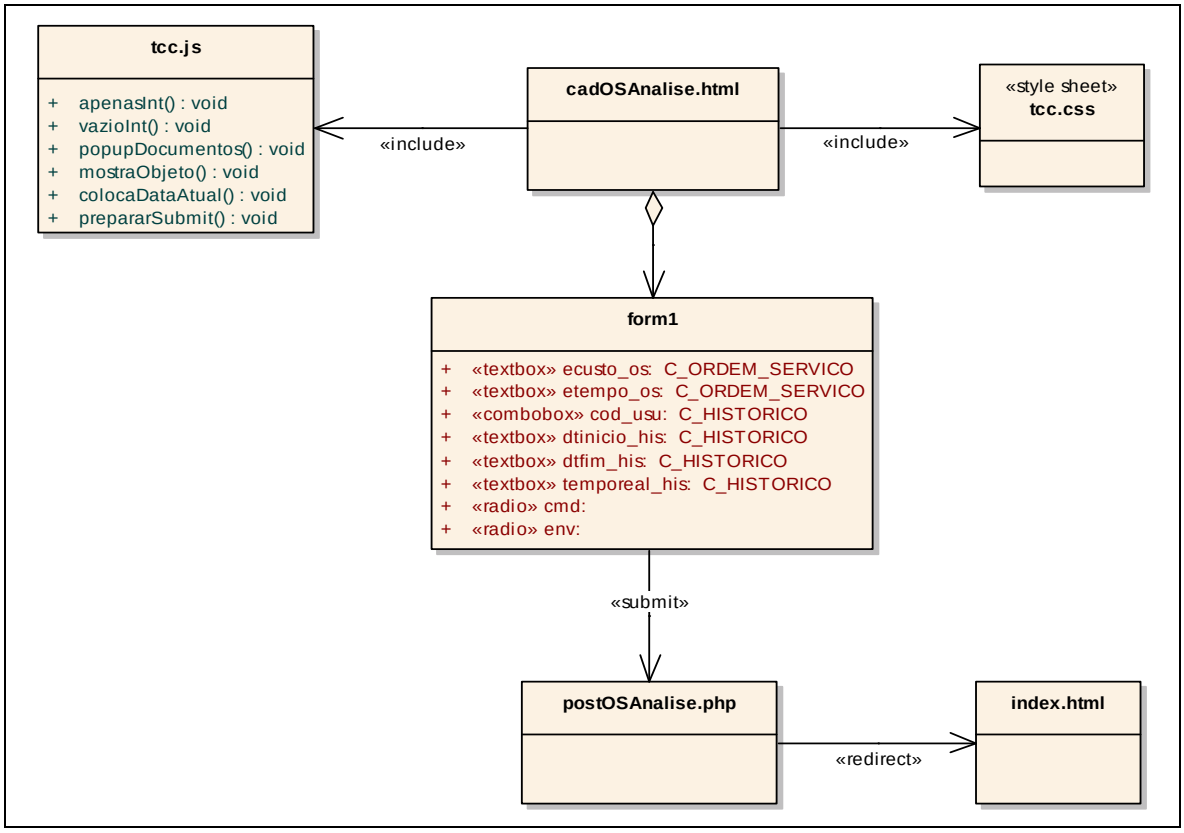


FIGURA 14 – Representação WAE do pacote análise

A figura 15 detalha o funcionamento do pacote de aprovação.

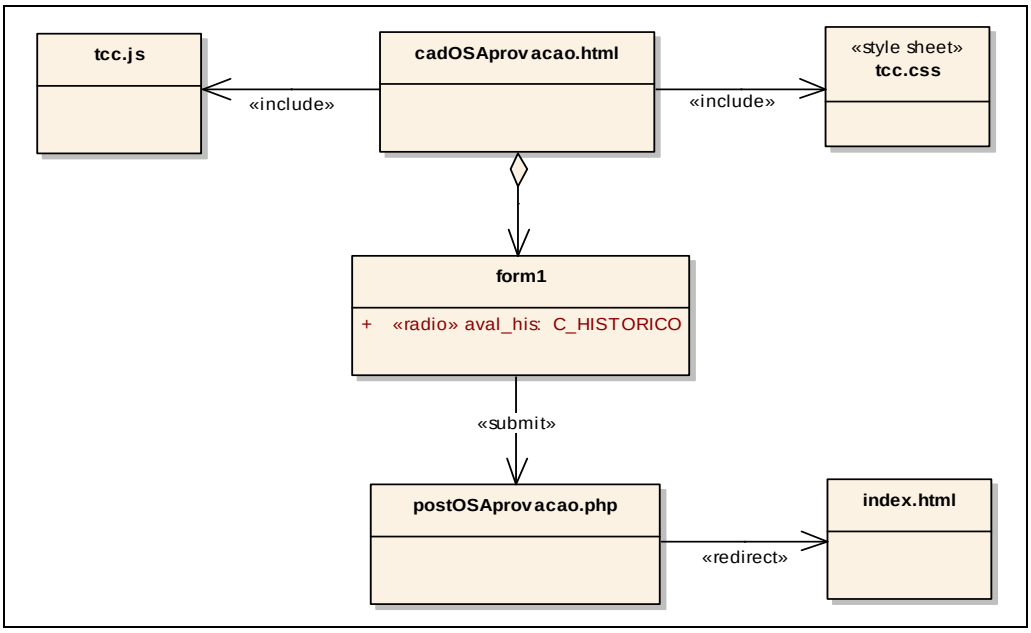


FIGURA 15 – Representação WAE do pacote aprovação

A figura 16 detalha o funcionamento do pacote de implementação.

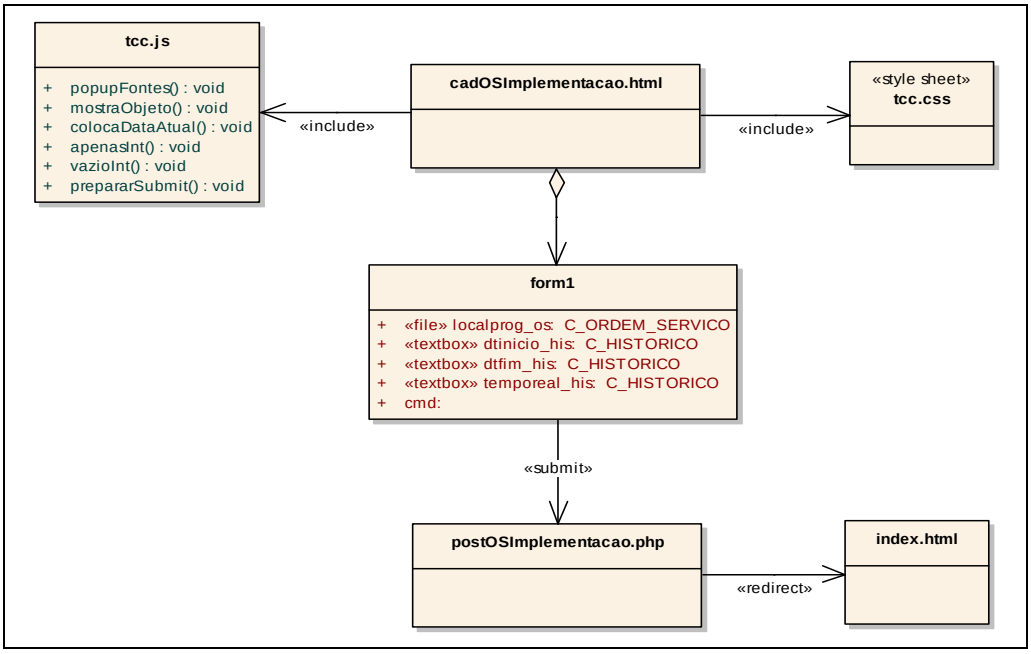


FIGURA 16 – Representação WAE do pacote implementação

A figura 17 detalha o funcionamento do pacote de aceitação e revisão.

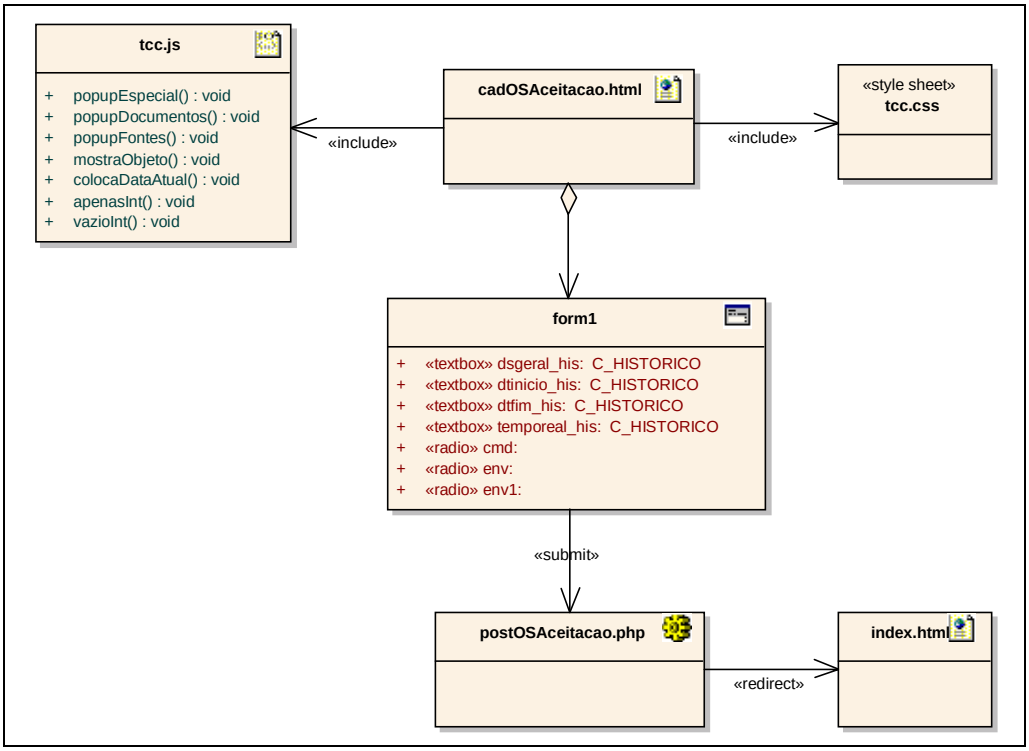


FIGURA 17 – Representação WAE do pacote aceitação

A figura 18 detalha o funcionamento do pacote de homologação.

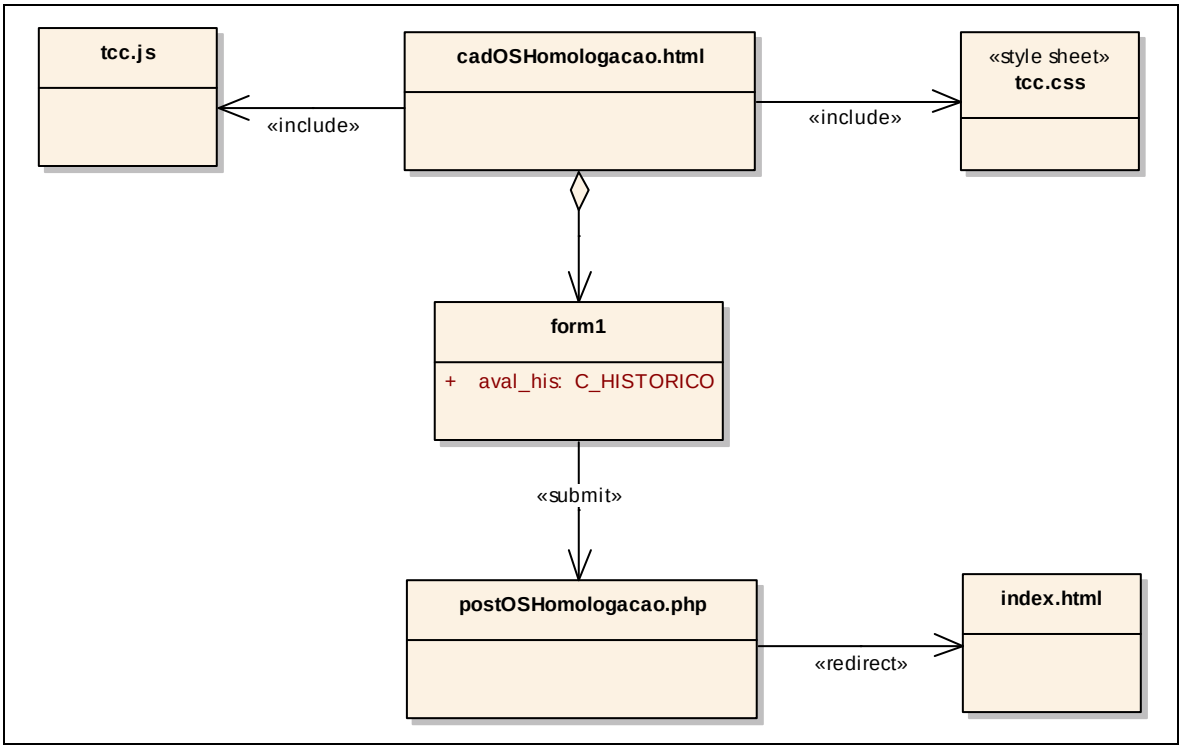


FIGURA 18 – Representação WAE do pacote homologação

3.2.5 DIAGRAMA ER

A figura 19 apresenta o modelo ER (Entidade e Relacionamento) utilizado pela ferramenta. A descrição do dicionário de dados encontra-se na seção Apêndice B.

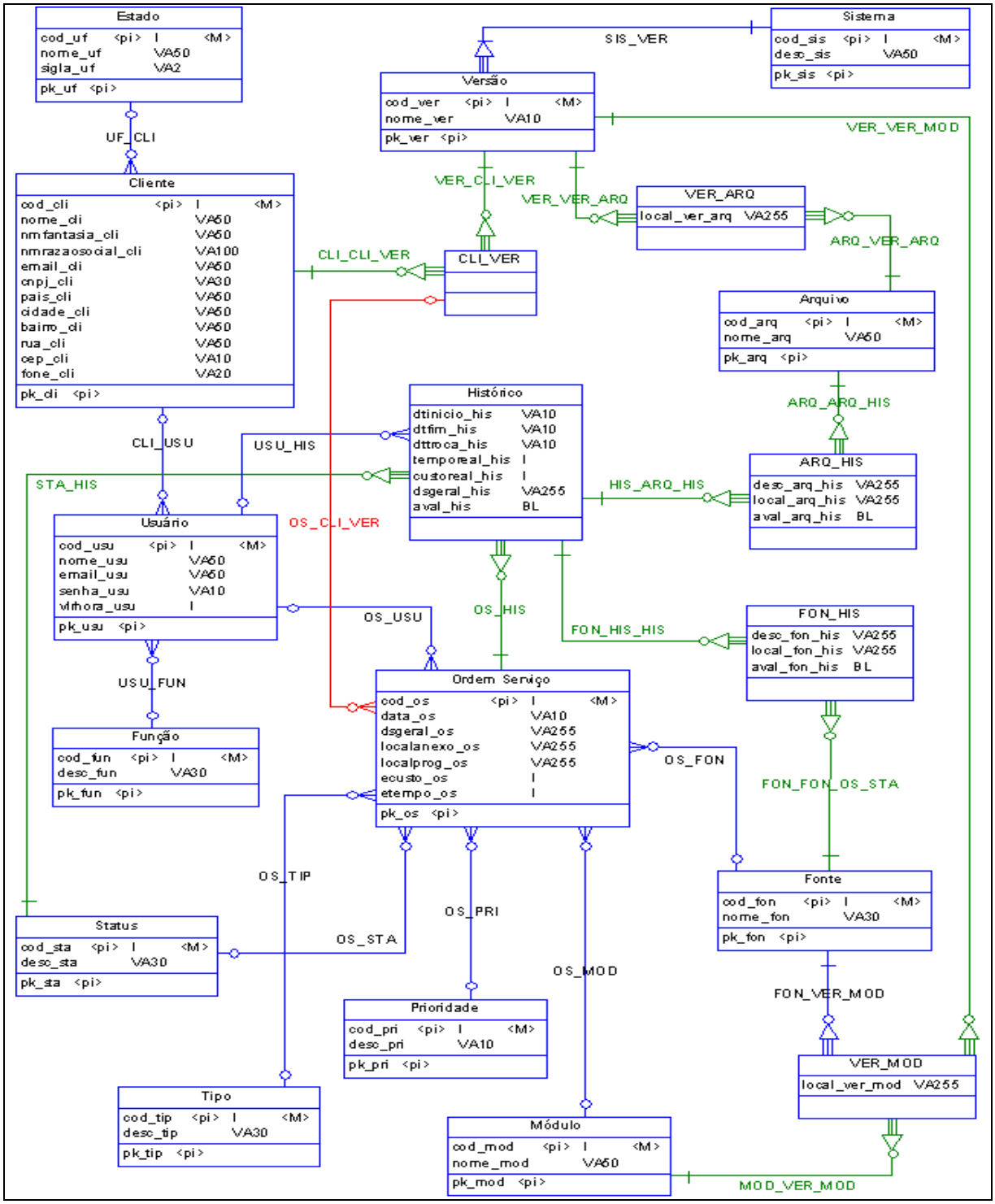


FIGURA 19 – Modelo ER Físico

3.3 IMPLEMENTAÇÃO

Esta seção contém o detalhamento sobre a implementação da ferramenta. O tópico inicial identifica as técnicas e ferramentas utilizadas. O tópico seguinte apresenta um estudo de caso do ponto de vista do usuário, destacando a funcionalidade ou operacionalidade da ferramenta. O último tópico descreve os resultados obtidos.

3.3.1 TÉCNICAS E FERRAMENTAS UTILIZADAS

A ferramenta implementada faz uso das tecnologias atuais para desenvolvimento de sistemas *web*, como o PHP 5.0 para a codificação e acesso ao banco de dados, o *Firebird 1.5* como SGDB (Sistema Gerenciador de Banco de Dados) e outros recursos como HTML (*Hipertext Markup Language*) , *Javascript* e CSS para a interface e navegação do sistema. Nota-se que todas as tecnologias utilizadas no desenvolvimento desta ferramenta são do tipo *Freeware* (Software Livre) ou *Open Source* (Código Aberto).

Hypertext Preprocessor (PHP) é uma linguagem de script que pode ser acoplada ao HTML, permitindo a construção de páginas *web* dinamicamente. Os scripts PHP são interpretados no servidor da aplicação *web*. A versão 5 do PHP utiliza uma nova *engine* denominada *Zend Engine II*. O manuseio de objetos nesta nova *engine* foi totalmente reescrito, possibilitando o uso de novos recursos em OO (Orientação a Objeto). (ZEND, 2004).

Cascading Style Sheets (CSS) é uma linguagem desenvolvida pela W3C (*World Wide Web Consortium*) que oferece um controle visual nas apresentações de páginas *web*. O CSS pode ser acoplado ao HTML, permitindo a inclusão de efeitos visuais baseado em eventos. São vários os navegadores e programas que suportam a linguagem. O CSS pode ser utilizado também em dispositivos móveis como telefones, PDAs (*Personal Digital Assistants*) e televisores à pilha. (W3C, 2004).

A figura 20 apresenta o arquivo CSS utilizado pela ferramenta.

```
.title { font-family: arial; font-size: 16px; font-weight: bolder; color: #ffffff; }
.caption { font-family: tahoma; font-size: 11px; text-align: right; }
.text { font-family: tahoma; font-size: 11px; }
.textLogin { font-family: tahoma; font-size: 11px; color: #ffffff; }
.titleFont { font-family: arial; font-size: 21px; text-decoration: none; color: #ffffff; text-align: left; }
.menuitem { font-family: verdana; font-size: 11px; text-decoration: none; color: #000080; padding: 3px 3px 3px 3px; }
.menuitem:hover { background-color: #c0c0c0; border: 1px solid #000000; color: #000000; padding: 2px 2px 2px 2px; }
.submenuitem { font-family: verdana; font-size: 11px; text-decoration: none; color: #ffffff; padding: 3px 3px 3px 3px; }
.submenuitem:hover { background-color: #c0c0c0; border: 1px solid #000000; color: #000000; padding: 2px 2px 2px 2px; }
.subMenu { background-color: #6a5acd; border-bottom: 1px solid #000000; color: #ffffff; padding: 5px 5px 5px 5px; }
.rowGrid { background-color: #ffffff; }
.rowGrid:hover { background-color: #fff8dc; }
.lineGrid { border-bottom: 1px solid #c0c0c0; }
.linkGrid { font-family: tahoma; font-size: 11px; text-decoration: none; color: #000080; }
.linkGrid:hover { color: #ff6347;}
.headerGrid { font-family: tahoma; font-size: 11px; color: #000080; }
.hum { background-image: url(imagens/hum.gif); background-repeat: repeat-x; padding: 0px 10px 0px 10px; }
.huml { background-image: url(imagens/huml.gif); background-repeat: no-repeat; }
.hum2 { background-image: url(imagens/hum2.gif); background-repeat: repeat-x; padding: 0px 10px 0px 10px; }
.normalitem { font-family: tahoma; font-size: 11px; text-decoration: none; color: #ffffff; }
.normalitem:hover { text-decoration: underline; }
.textAuthor { font-family: tahoma; font-size: 11px; color: #ffffff; padding: 5px 5px 5px 5px; }
```

FIGURA 20 – Arquivo CSS utilizado pela ferramenta

Antes de realmente explicar as técnicas utilizadas na implementação da ferramenta, é importante descrever sua estrutura. A estrutura da ferramenta está representada na figura 21.

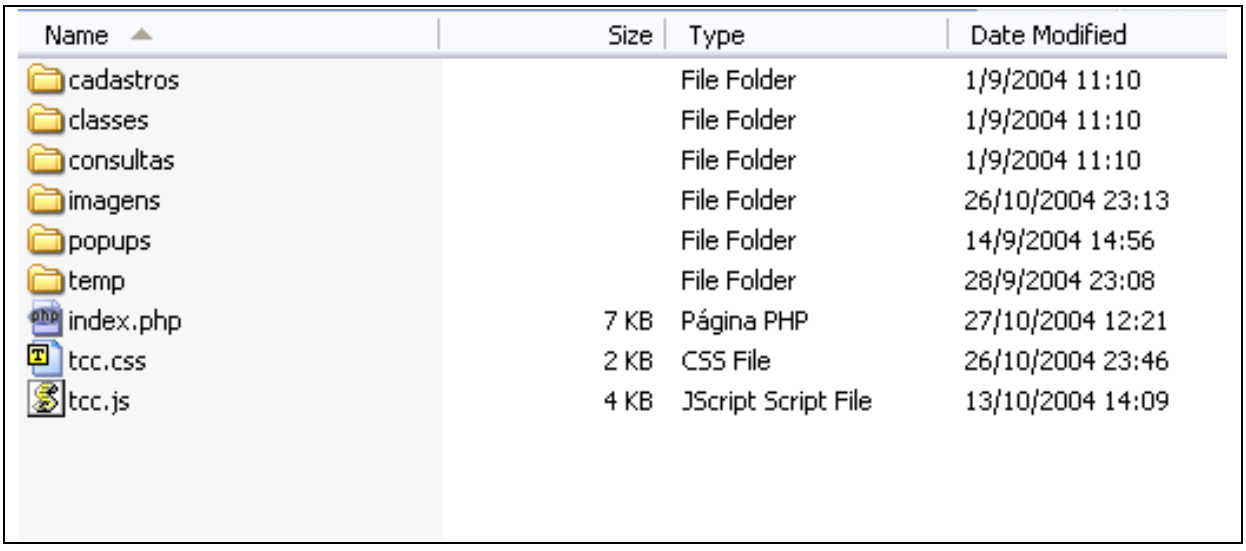


FIGURA 21 – Estrutura da ferramenta

Nota-se a existência de três arquivos na pasta principal. São arquivos do tipo “php”, “css” e “js”. Estes arquivos referem-se respectivamente à página de entrada, ao padrão visual adotado e à navegação e validação dos dados.

Verifica-se também, que a estrutura final foi dividida em seis pastas distintas. Cada pasta armazena arquivos correspondentes a seus nomes exceto a pasta ”temp”, cuja finalidade é armazenar os anexos da manutenção.

Outra característica da ferramenta desenvolvida é que a mesma faz uso de POO (Programação Orientada a Objetos). As classes implementadas correspondem a uma tabela do *database*, exceto a super-classe, onde existem funcionalidades de controle e acesso ao banco de dados (maiores detalhes na figura 22).

A figura 22 representa o comportamento da classe SISTEMA. A classe C_INTERBASE é a super-classe. Esta classe contém os métodos de controle e acesso ao banco de dados. A classe INTERBASE implementa as funcionalidades extras da super-classe. As classes C_SISTEMA e SISTEMA representam a tabela SISTEMA, situada na seção 3.2.4. A classe C_SISTEMA contém os métodos de inserção, atualização, remoção e consulta aos dados, enquanto que a classe SISTEMA possui funcionalidades personalizadas. As figuras 23-26 mostram uma parte da implementação das classes existentes na figura 22.

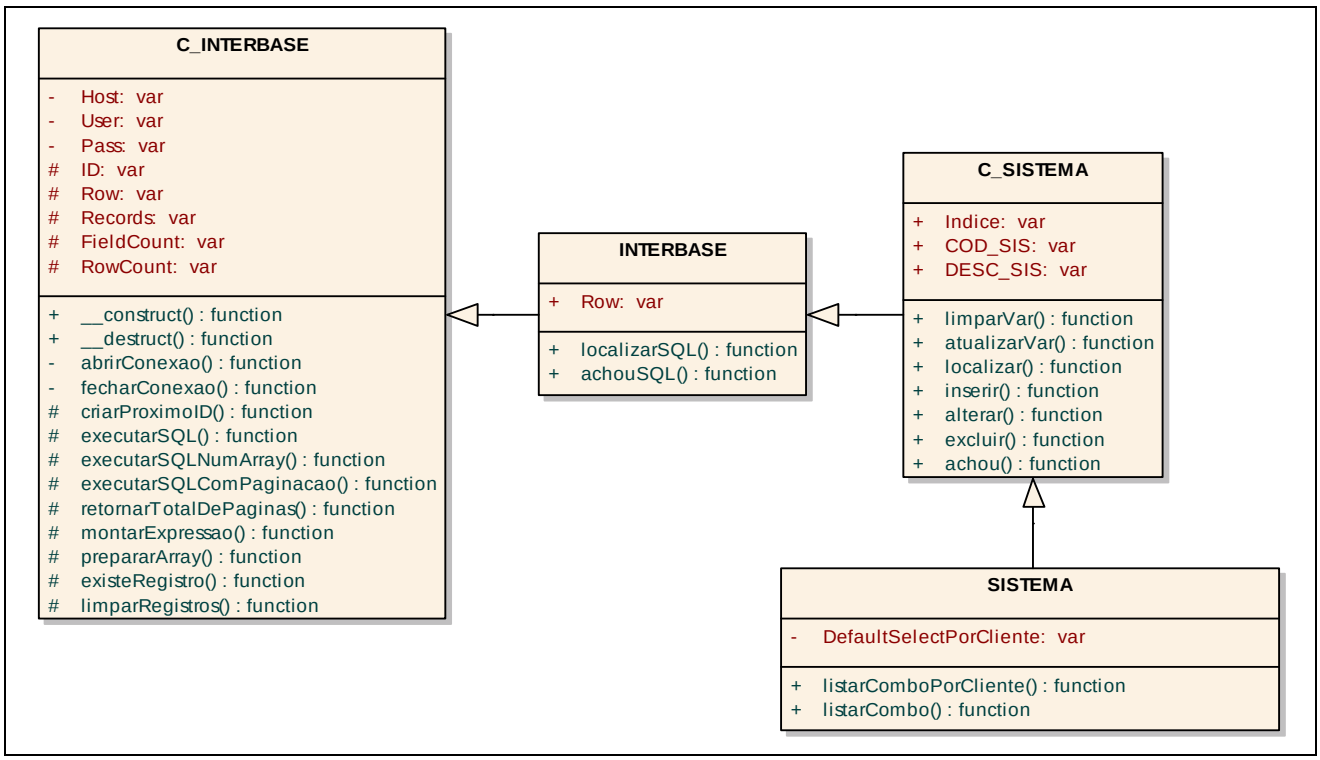


FIGURA 22 – Comportamento da classe SISTEMA

```php
<?php

class C_INTERBASE {

private $Host = "localhost:C:\Documents and Settings\Leonardo\My Documents\TCC\Diagramas\TCC.GDB";
private $User = "SYSDBA";
private $Pass = "masterkey";
protected $ID;
protected $Row;
protected $Records;
protected $FieldCount;
protected $RowCount;

public function __construct() {
    $this->abrirConexao();
}

public function __destruct() {
    $this->fecharConexao();
}

private function abrirConexao() {
    $this->ID = ibase_pconnect($this->Host,$this->User,$this->Pass) or die(ibase_error());
}

private function fecharConexao() {
    ibase_close($this->ID);
}

protected function criarProximoID($table,$field) {
    $this->Records = ibase_query($this->ID,"select max(".$field.") ultimo from ".$table) or die(ibase_error());
    if ($this->existeRegistro())
        return $this->Row->ULTIMO + 1;
}
```

FIGURA 23 – Implementação da super-classe C_INTERBASE

```php
<?php

include_once("C_INTERBASE.php");

class INTERBASE extends C_INTERBASE {

public $Row;

public function localizarSQL($sql) {
    $sql = trim($sql);
    if (strtoupper($sql[0]) == 'S')
        $this->executarSQL($sql);
}

public function achouSQL() {
    return ($this->existeRegistro());
    $this->Row = parent::Row;
}

}

?>
```

FIGURA 24 – Implementação da classe INTERBASE

```php
<?php

include_once("INTERBASE.php");

class C_SISTEMA extends INTERBASE {

public $Indice;
public $COD_SIS;
public $DESC_SIS;

public function limparVar() {
    $this->COD_SIS = null;
    $this->DESC_SIS = null;
}

public function atualizarVar() {
    $this->COD_SIS = $this->Row->COD_SIS;
    $this->DESC_SIS = $this->Row->DESC_SIS;
}

public function localizar() {
    if (empty($this->Indice))
        $this->Indice = "COD_SIS";
    $S = $this->montarExpressao(array("COD_SIS","DESC_SIS"),array($this->COD_SIS,$this->DESC_SIS));
    $this->executarSQL("select * from SISTEMA ".$S." order by ".$this->Indice);
}

public function inserir() {
    $this->COD_SIS = $this->criarProximoID('SISTEMA','COD_SIS');
    $this->executarSQL("insert into SISTEMA values (".$this->COD_SIS.", '".$this->DESC_SIS."')");
}

public function alterar() {
    $this->executarSQL("update SISTEMA set DESC_SIS = '".$this->DESC_SIS."' where COD_SIS = ".$this->COD_SIS." ");
}

public function excluir() {
    $this->executarSQL("delete from SISTEMA where COD_SIS = ".$this->COD_SIS." ");
}
```

FIGURA 25 – Implementação da classe C_SISTEMA

```php
<?php

include_once("C_SISTEMA.php");

class SISTEMA extends C_SISTEMA {

private $DefaultSelectPorCliente =
"select
    s.cod_sis,
    s.desc_sis
from
    sistema s,
    cli_ver cv
where
    s.cod_sis = cv.cod_sis and
    cv.cod_cli = %cod_cli%
order by
    s.desc_sis;
";

public function listarComboPorCliente($cliente = 0,$posicao = '') {
    $S = '';
    $this->limparVar();
    $this->localizarSQL(str_replace("%cod_cli%",$cliente,$this->DefaultSelectPorCliente));
    while ($this->achouSQL()) {
        if ($this->Row->COD_SIS == $posicao)
            $S .= "<option value='".$this->Row->COD_SIS."' selected>".$this->Row->DESC_SIS."</option>\n";
        else
            $S .= "<option value='".$this->Row->COD_SIS."'>".$this->Row->DESC_SIS."</option>\n";
    }
    return $S;
}

public function listarCombo($posicao = '') {
    $S = '';
    $this->limparVar();
    $this->localizar();
    while ($this->achou()) {
        if ($this->COD_SIS == $posicao)
            $S .= "<option value='".$this->COD_SIS."' selected>".$this->DESC_SIS."</option>\n";
        else
            $S .= "<option value='".$this->COD_SIS."'>".$this->DESC_SIS."</option>\n";
    }
    return $S;
}
```

FIGURA 26 – Implementação da classe SISTEMA

Com o objetivo de facilitar a manutenção desta ferramenta, foi criado pelo autor deste trabalho um programa auxiliar em *Delphi 5.0* que gera código em PHP 5.0 ou superior. O código gerado refere-se as classes com as iniciais "C_", cuja implementação foi apresentada anteriormente. Uma amostra da interface do programa:

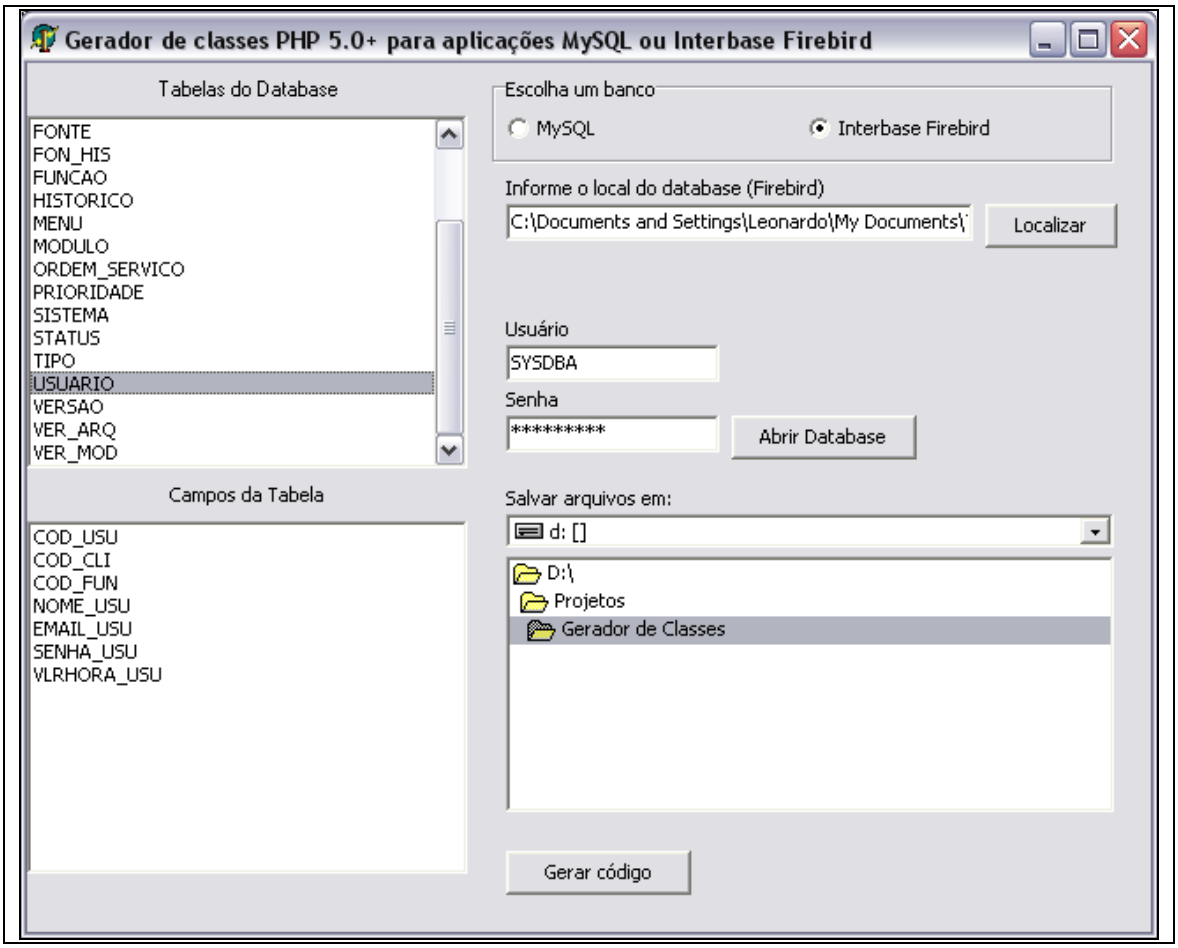


FIGURA 27 – Interface do programa auxiliar

Observa-se que o programa auxiliar desenvolvido, gera código a partir de *databases*. Os bancos de dados suportados no momento são o *MySQL* e o *Firebird*. Outra característica deste programa é que o mesmo gera uma classe para cada tabela, contendo as operações básicas de inserção, alteração, remoção e consulta aos dados. Com o programa auxiliar, foi possível manter as classes sempre atualizadas com o Modelo ER, agilizando o processo de implementação, como também, criando uma padronização de código, visando facilitar a manutenção da ferramenta.

3.3.2 OPERACIONALIDADE DA IMPLEMENTAÇÃO

Esta seção apresenta um estudo de caso, do ponto de vista do usuário, objetivando mostrar a funcionalidade e operacionalidade da ferramenta. O estudo de caso abrange o *workflow* proposto pelo Diagrama de Atividades, apresentado na seção 3.2.1., e descreve o processo em um sistema hoteleiro.

Um hotel localizado no litoral de Santa Catarina, há tempos utiliza o Sistema Hoteleiro fornecido pela LRW Systems. Porém, devido às novas tendências do mercado, o hotel procurava intensamente um diferencial. Assim surgiu a idéia de dar pontos de bonificação ao hóspede para cada dia de estadia. O hotel gostaria de ver sua idéia incorporada ao Sistema Hoteleiro. Para isso, o Solicitante executou a atividade "ATV 01. Solicitação de manutenção", preenchendo os campos de acordo com a figura 28.

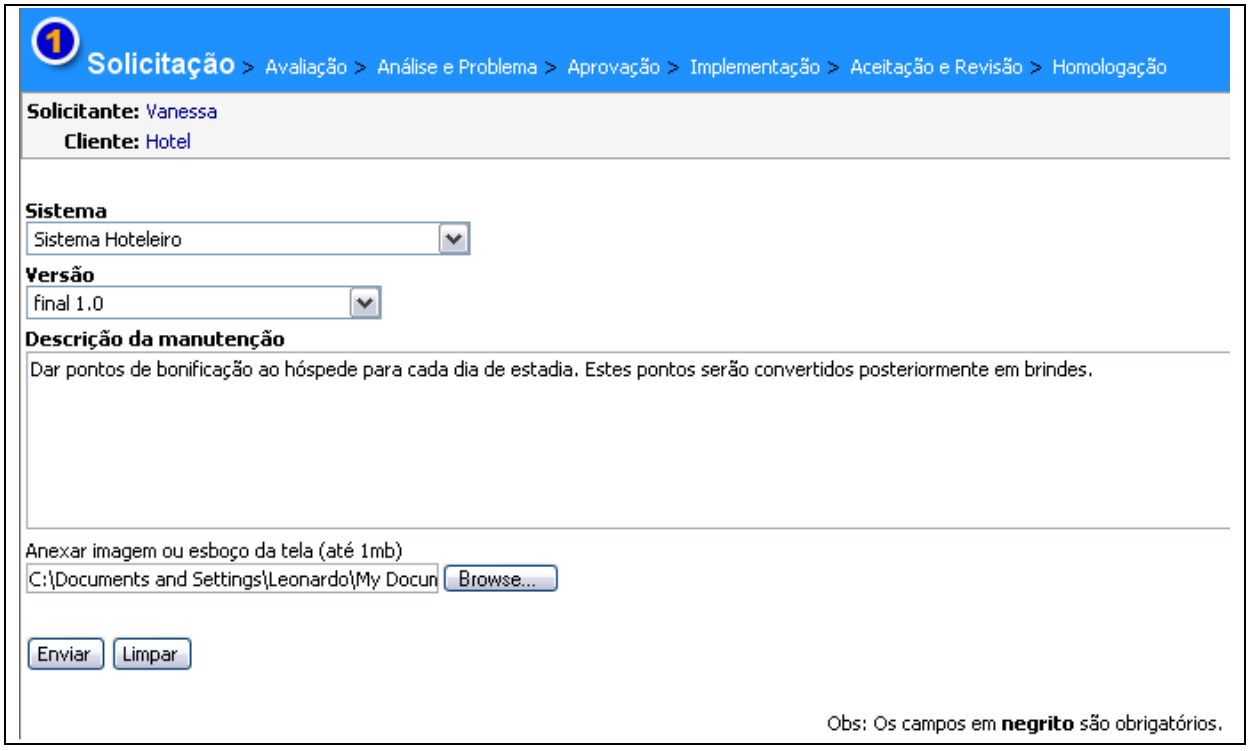


FIGURA 28 – Solicitação da manutenção

Após a confirmação da solicitação pelo Solicitante, a ferramenta encaminha a manutenção para o Coordenador. O Coordenador executa as atividades "ATV 02. Avaliação da manutenção" e "ATV 03. Definir analista", preenchendo os campos de acordo com a figura 29.

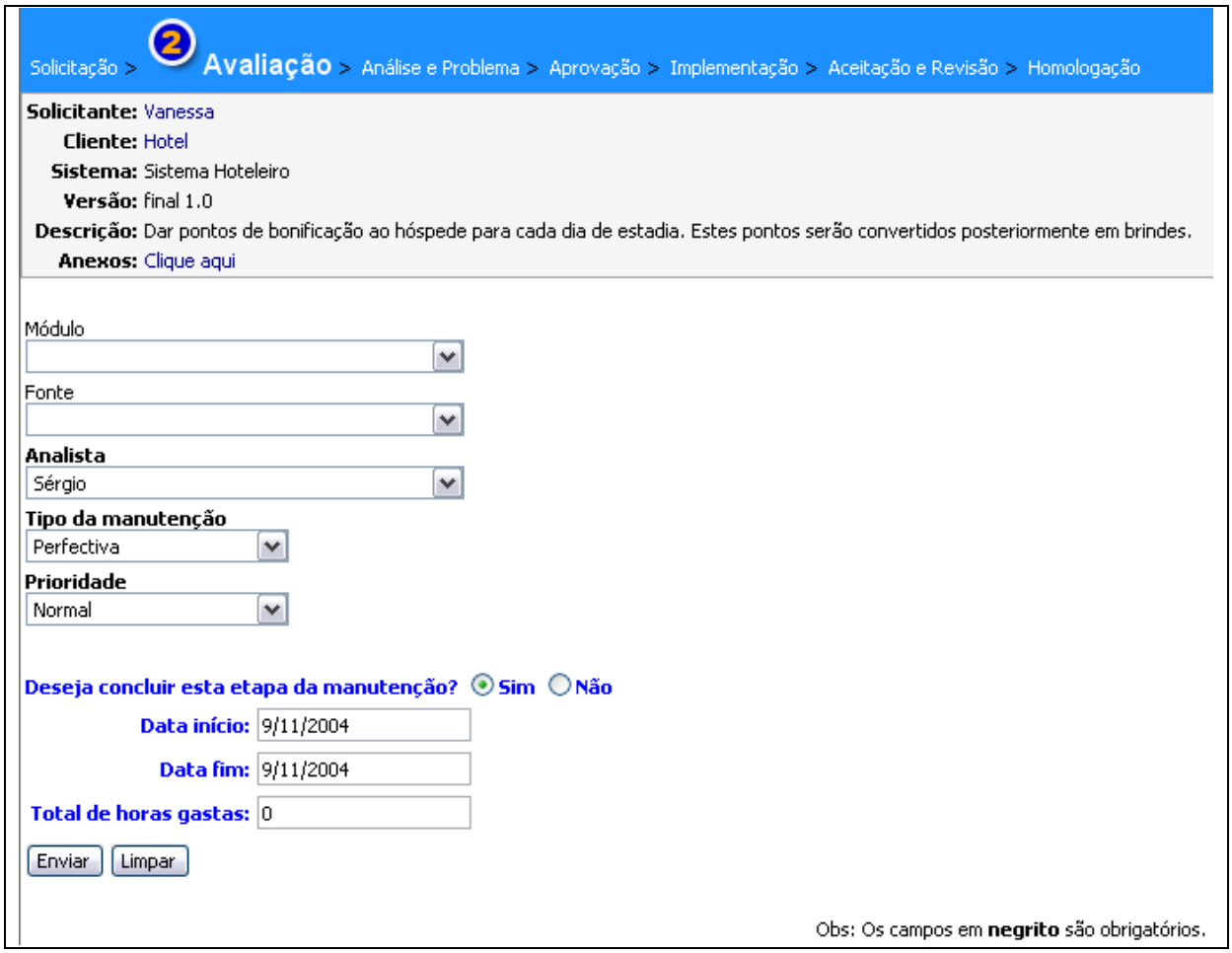


FIGURA 29 – Avaliação da manutenção

Após a confirmação da avaliação pelo Coordenador, a ferramenta encaminha a manutenção para o Analista. O Analista executa as atividades “ATV 04. Análise e Problema da manutenção”, “ATV 05. Determinar estimativa de custo”, “ATV 07. Atualizar documentos envolvidos” e “ATV 08. Definir programador”, preenchendo os campos de acordo com a figura 30.

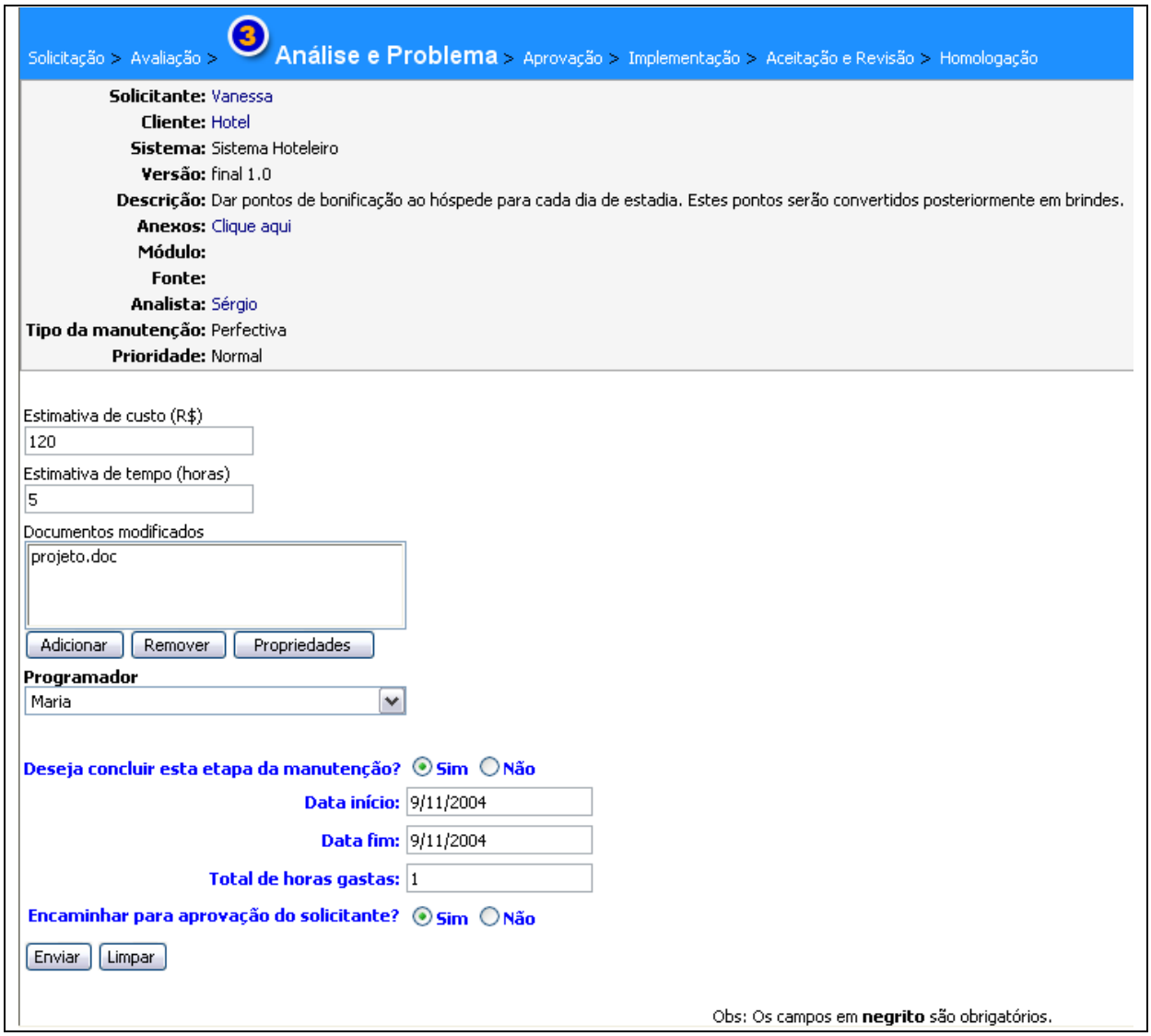


FIGURA 30 – Análise e Problema da manutenção

Após a confirmação desta etapa pelo Analista, a ferramenta encaminha a manutenção para o Solicitante. O Solicitante executa a atividade “ATV 06. Aprovação da manutenção”, preenchendo os campos de acordo com a figura 31.

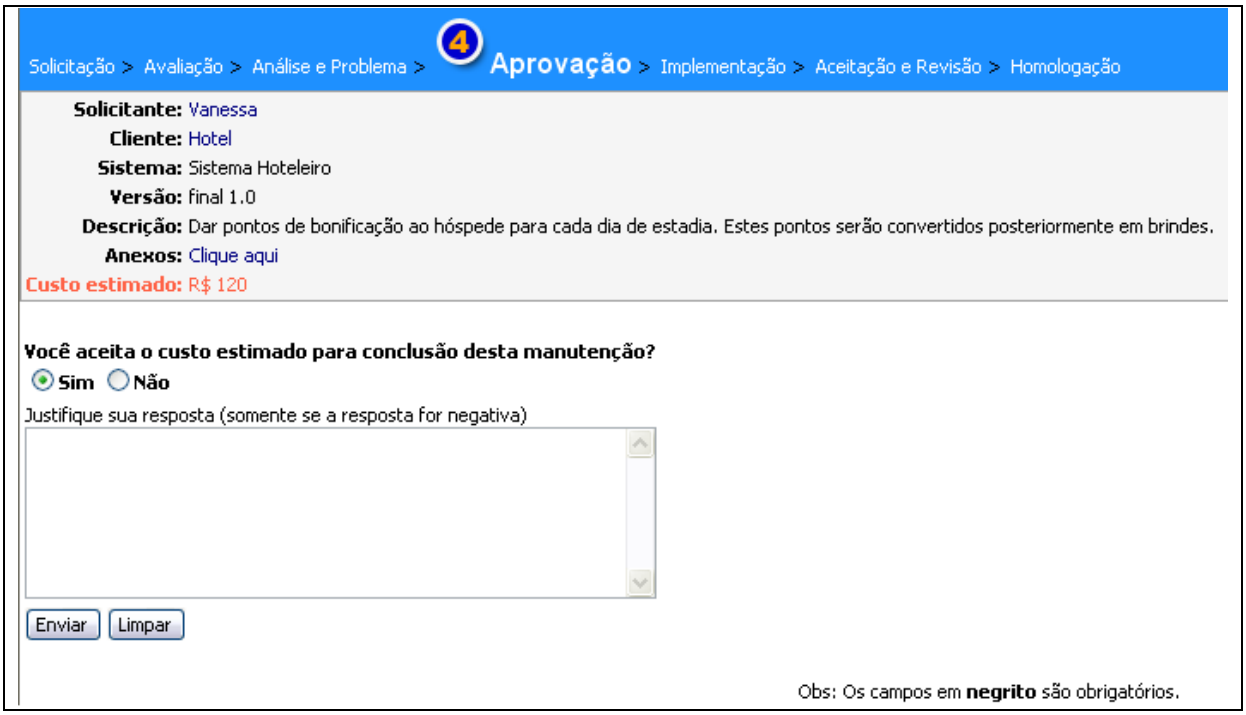


FIGURA 31 – Aprovação da manutenção

Após a confirmação da aprovação pelo Solicitante, a ferramenta encaminha a manutenção para o Analista. O Analista faz a revisão do que foi feito na etapa de análise, conforme mostra a figura 32.

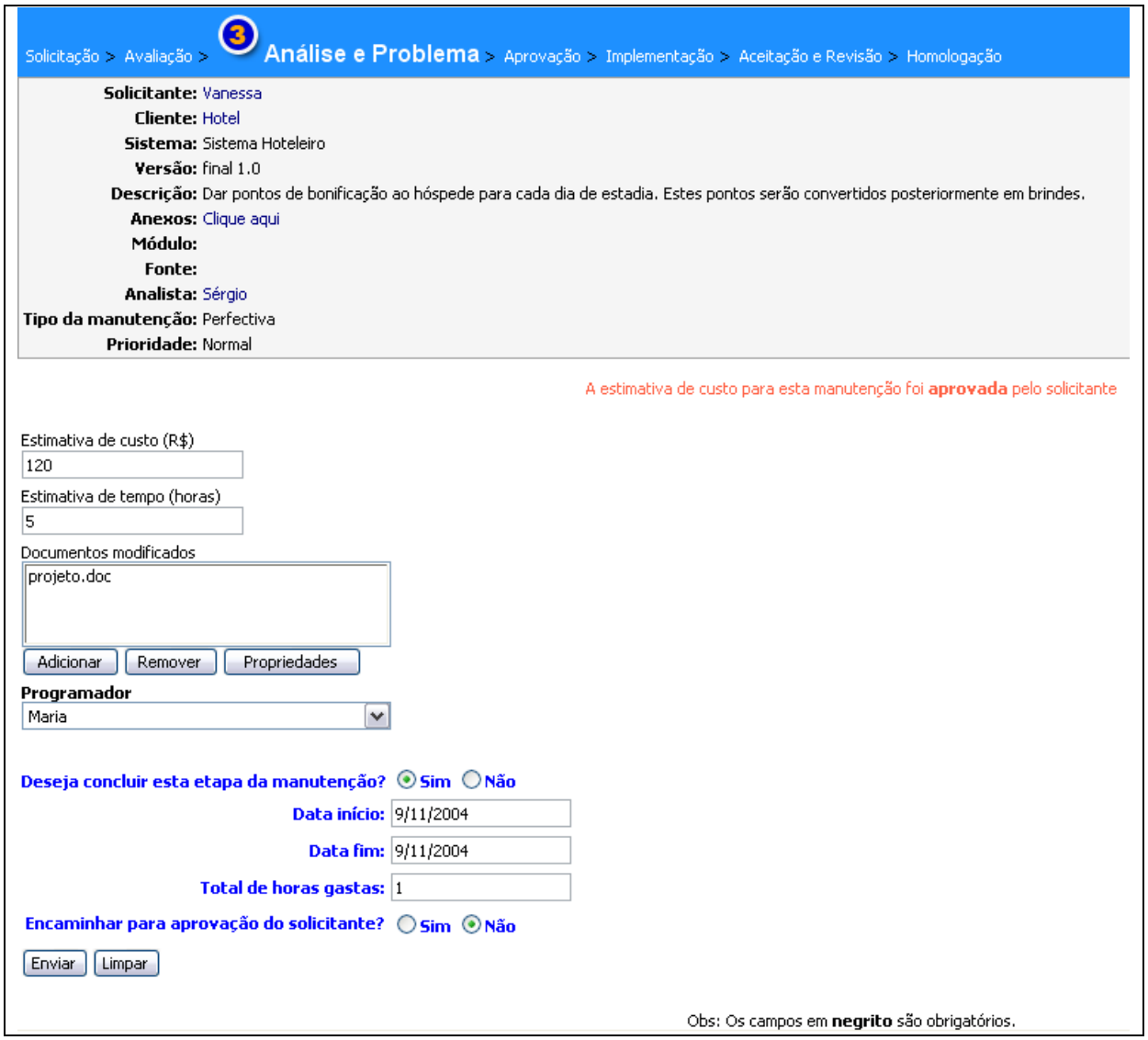


FIGURA 32 – Análise e Problema (estimativa de custo aprovada)

Após a confirmação desta etapa pelo Analista, a ferramenta encaminha a manutenção para o Programador. O Programador executa as atividades “ATV 09. Implementação da manutenção”, “ATV 10. Atualizar fontes envolvidos” e “ATV 11. Anexar instalador”, preenchendo os campos de acordo com a figura 33.

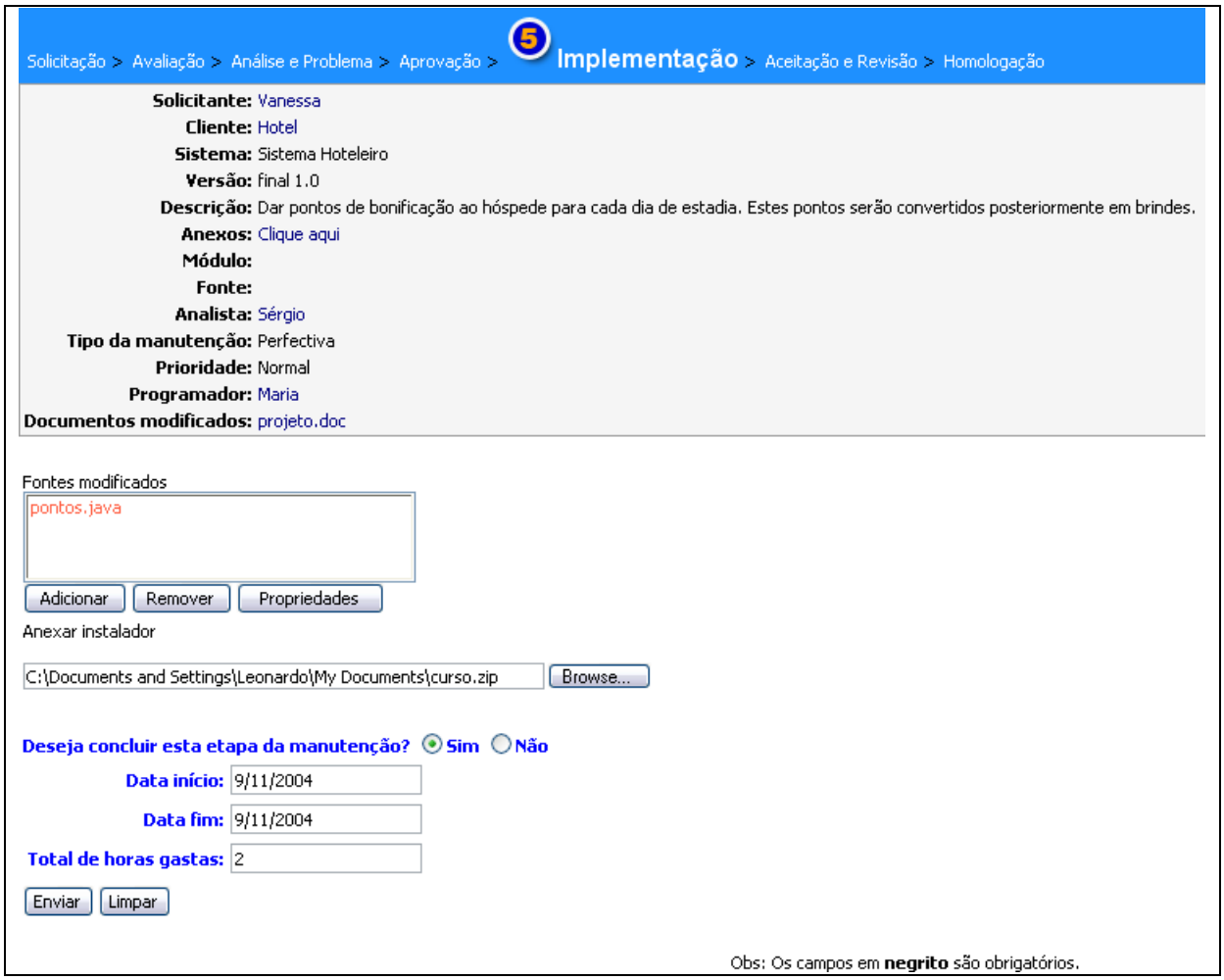


FIGURA 33 – Implementação da manutenção

Após a confirmação da implementação pelo Programador, a ferramenta encaminha a manutenção para o Analista. O Analista executa a atividade “ATV 12. Aceitação e Revisão da manutenção”, preenchendo os campos de acordo com a figura 34.

FIGURA 34 – Aceitação e Revisão da manutenção

Após a confirmação da revisão pelo Analista, a ferramenta encaminha a manutenção para o Solicitante. O Solicitante executa a atividade “ATV 13. Homologação da manutenção”, preenchendo os campos de acordo com a figura 35.

FIGURA 35 – Homologação da manutenção

Após a confirmação da homologação pelo Solicitante, a ferramenta finaliza a manutenção. O Sistema Hoteleiro foi atualizado com sucesso pela equipe de manutenção da LRW Systems.

As demais funcionalidades implementadas para viabilizar o *workflow*, consideradas apenas como funcionalidades de suporte ao processo, estão ilustradas no Apêndice C.

3.4 RESULTADOS

O quadro 3 apresenta um comparativo entre as etapas e procedimentos propostos pela ISO/IEC 14764, identificados na seção 2.3.3, e as funcionalidades implementadas pela CASE deste trabalho. A primeira coluna refere-se às etapas de manutenção existentes no modelo da ISO. A segunda coluna contém os procedimentos da etapa. A terceira coluna define se são procedimentos de entrada ou saída. E a última coluna identifica se o procedimento está presente na ferramenta implementada e qual a etapa/funcionalidade se refere.

Etapas ISO/IEC 14764	Procedimentos da etapa	Controle	CASE proposta
Execução do Processo	*baseline*[1]	Entrada	Na etapa de Análise é registrado a estimativa do tempo total, servindo como *baseline* para o pedido.
	documentação do sistema	Entrada	O coordenador e analista podem efetuar o cadastro de documentação dos sistemas em manutenção.
	SM ou RP	Entrada	Atendido através do cadastro de solicitação de manutenção pelo solicitante.
	estratégia da manutenção	Saída	Considera-se neste item a execução do *workflow* proposto pela ferramenta.
	procedimentos da manutenção	Saída	
	procedimentos para definição do problema	Saída	Na etapa de avaliação o coordenador identifica o problema categorizando em um tipo de manutenção específico.
	feedback para os usuários	Saída	Acompanhamento do pedido pelo solicitante.
	gerência de configuração[2]	Saída	Refere-se ao próprio *workflow* que permite gerenciar o processo e efetuar o acompanhamento
Análise e Problema	SM ou RP	Entrada	Em todas as etapas há o relato completo (cabeçalho do pedido) permitindo acompanhamento.
	baseline atualizada	Entrada	É registrado o tempo total gasto em cada atividade.
	repositório do software	Entrada	É permitido o *upload* e *download* dos fontes, ou seja, internamente há um repositório. Embora recomenda-se a implementação de uma ferramenta para gerenciamento específico (detalhes na seção 4.1)
	documentação do sistema	Entrada	O coordenador e analista podem efetuar o cadastro de documentação dos sistemas em manutenção.
	análise do impacto	Saída	Sim, nesta etapa existem alguns campos que abrangem a análise de impacto como tempo e custos envolvidos
	solução recomendada	Saída	Sim, a solução recomendada é anexada na manutenção
	aprovação da manutenção	Saída	Sim, na etapa de aprovação o solicitante deve aprovar o orçamento da manutenção

[1] Tempo de duração previsto para um item de configuração (ISO/IEC 14764)

[2] atividade abrangente que é aplicada em todo o processo de engenharia de software. Uma vez que uma mudança pode ocorrer a qualquer tempo, as atividades de gerenciamento de configuração são desenvolvidas para identificar mudança; controlar a mudança; garantir que mudança esteja sendo adequadamente implementada; e relatar a mudança a outras pessoas que possam ter interesse nela. (PRESSMAN, 1995)

	documentação atualizada	Saída	Durante a execução do *workflow* é permitido atualização da documentação na etapa de Análise
Implementação	*baseline* atualizada	Entrada	È registrado o tempo total gasto na etapa de Implementação, como também nas demais atividades.
	SM ou RP aprovada	Entrada	Em todas as etapas há o relato completo (cabeçalho do pedido) permitindo acompanhamento.
	documentação atualizada	Saída	O *workflow* não permite a atualização de documentos na etapa de implementação, apenas na etapa de Análise.
	fontes modificados	Saída	Na etapa de Implementação a ferramenta permite o *upload* dos fontes atualizados.
	resultado dos testes	Saída	Não é prevista alteração da documentação na etapa de implementação. No entanto, na etapa de Aceitação o analista irá registrar o resultado dos testes realizados.
Aceitação e Revisão	software modificado	Entrada	Na etapa de Aceitação a ferramenta permite o *download* dos fontes.
	resultado dos testes	Entrada	Não foi previsto documentação dos testes por parte da equipe de implementação.
	nova *baseline* com as modificações aceitas	Saída	Não é apresentado atualmente pela ferramenta o tempo total para uma modificação. Embora sejam armazenados dados para este acompanhamento.
	modificações rejeitadas	Saída	O analista registra comentários referente aos itens reprovados.
	aval de aceitação da manutenção	Saída	A ferramenta implementa este procedimento na etapa de homologação, onde o solicitante aprova as modificações.
	resultado dos testes	Saída	Refere-se a total aprovação dos itens pelo analista ou relato dos itens reprovados.
Migração	ambiente antigo	Entrada	No *workflow* proposto a migração é considerada como uma manutenção adaptativa (devendo ser identificada pelo coordenador) e, portanto, seguirá o mesmo fluxo já descrito. Não foi feito um procedimento específico para esta categoria de manutenção.
	ambiente novo	Entrada	
	baseline antiga	Entrada	
	baseline nova	Entrada	
	plano de migração	Saída	
	software migrado	Saída	
	notificação de conclusão	Saída	
	backup dos produtos e dados antigos	Saída	
Encer	software antigo	Entrada	Não foi implementado nenhuma funcionalidade específica para tratar da retirada do produto do mercado. Apenas, auxilia no registro das versões disponibilizadas
	software novo	Entrada	
	ambiente antigo	Entrada	
	ambiente novo	Entrada	

Ortemar	plano de retirada do software	Saída	e os seus clientes.
	software retido	Saída	
	notificação de conclusão	Saída	
	backup do software antigo	Saída	

QUADRO 3 – Resultados alcançados com a implementação da CASE

4 CONCLUSÕES

A CASE implementada gerencia o trabalho das equipes de manutenção de software mantendo a documentação dos sistemas, armazenando o histórico de cada manutenção, controlando as atividades da equipe e facilitando a comunicação entre os membros da equipe e o cliente. Sendo assim, a ferramenta atende a todos os objetivos propostos na seção 1.1.

O modelo de manutenção adotado pela ferramenta segue os padrões propostos pela ISO/IEC 14764, identificados na seção 2.3.3. Existem seis etapas distintas no modelo de manutenção da ISO/IEC 14764. A CASE resultante deste trabalho contempla principalmente as etapas de análise e problema, implementação e aceitação e revisão.

A etapa de execução do processo, existente no modelo de manutenção da ISO/IEC 14764, foi integrada na CASE, somando as funcionalidades das etapas de solicitação e avaliação da manutenção. As etapas de migração e encerramento, existentes na ISO/IEC 14764, não ficaram evidentes na ferramenta. Contudo, o modelo de manutenção empregado na CASE, permite a execução destas etapas de forma satisfatória.

Com relação às tecnologias empregadas na CASE, destaca-se o uso do PHP 5, com lançamento recente, cujo modelo de objetos foi totalmente reescrito, possibilitando o uso de POO. Com o uso de POO, foi possível estabelecer um padrão de desenvolvimento e facilitar a manutenção da ferramenta. Outra tecnologia que auxiliou no processo de implementação da CASE foi o programa auxiliar identificado na seção 3.3.1.

A ferramenta desenvolvida apresenta algumas deficiências ou limitações, principalmente na parte de segurança aplicada aos produtos dos sistemas, como fontes e documentos. Além disso, não se implementou controles para o uso compartilhado de códigos fontes e documentação, pois o foco principal da CASE é o processo de manutenção.

4.1 EXTENSÕES

Devido às deficiências encontradas no que se refere ao compartilhamento de código, uma sugestão para trabalhos futuros seria a implementação de novos recursos para manter os produtos do software mais seguros. Uma idéia inicial seria a criação de um controle de fontes, ou integração com algum programa já existente, tal como o CVS (*Concurrent Versions System*) ou *Microsoft Visual Sourcesafe* que possuem essa funcionalidade.

Uma outra sugestão para implementações futuras seria a inclusão de gráficos no nível gerencial. Gráficos que permitam avaliar o desempenho, identificar deficiências ou apresentar as áreas mais dispendiosas da manutenção. A finalidade principal desses gráficos estaria voltada a auxiliar a equipe de manutenção na tomada de decisão.

Outra extensão proposta para esta ferramenta seria o estudo e implementação do padrão de projeto *Bridge* proposto por Gamma (2000), visando melhorar a manutenção e reutilização de código orientado a objetos.

REFERÊNCIAS BIBLIOGRÁFICAS

CONALLEN, Jim. **Desenvolvimento aplicações web com UML**. Rio de Janeiro: Campus, 2003.

GAMMA, Erich; HELM, Richard; JOHNSON, Ralph; VLISSIDES, John. **Padrões de projeto** : soluções reutilizáveis de software orientado a objetos. Porto Alegre : Bookman, 2000.

HOPPE, Charles; GRAHL, Everaldo Artur. **Software de apoio à manutenção de sistemas baseado em normas de qualidade**. 1999. Monografia (Trabalho de Conclusão de Curso em Ciências da Computação) – Departamento de Sistemas e Computação, Universidade Regional de Blumenau, Blumenau.

PETER, James F. e PEDRYCZ, Witold. **Engenharia de Software teoria e prática**. Rio de Janeiro: Campus, 2001.

PRESSMAN, Roger S. **Engenharia de software**. São Paulo: Makron Books, 1995.

PRESSMAN, Roger S. **Engenharia de software**. 5.ed. Rio de Janeiro: McGraw-Hill, 2002.

SCUSSIATO, Edédio; GRAHL, Everaldo Artur. **Processo de manutenção de sistema baseado na norma ISO/IEC 12207**. 1998. Monografia (Pós-Graduação ao nível de especialização em Tecnologia de Desenvolvimento de Sistemas) – Departamento de Sistemas e Computação, Universidade Regional de Blumenau, Blumenau.

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

ISO/IEC 14764 – INTERNATIONAL ORGANIZATION FOR STANDARDIZATION – INTERNATIONAL ELECTROTECHNICAL COMMISSION – **ISO/IEC 14764**: software engineering – software maintenance. Suíça, 1999.

W3C – WORLD WIDE WEB CONSORTIUM – **W3C**: style activity statement, Massachusetts, 1994. Disponível em <http://www.w3.org/Style/Activity>. Acesso em 12 nov. 2004.

ZEND – THE PHP COMPANY – **ZEND**: changes in PHP 5/Zend Engine II, Cupertino, 1999. Disponível em <http://www.zend.com/php5/articles/engine2-php5-changes.php>. Acesso em 10 nov. 2004.

APÊNDICE A – Descrição dos cenários

UC 01.01. Efetuar login no sistema

Cenários

Efetuar login no sistema {Principal}.
1) Usuário informa e-mail.
2) Usuário informa senha.
3) Sistema valida login.
4) Sistema executa UC 01.02. Consultar manutenções.

Login inválido {Exceção}.
No passo 3, caso o e-mail ou senha não confiram com os informados no sistema, notificar usuário do ocorrido.

UC 01.02. Consultar manutenções

Restrições

- *Pré-condição* . Usuário logado no sistema (UC 01.01).

Cenários

Consultar manutenções {Principal}.
1) Sistema filtra manutenções pendentes para o usuário logado.
2) Sistema mostra resultado.

Não existem manutenções {Exceção}.
1) No passo 2, pode não existir resultado. Notificar usuário do ocorrido.

UC 01.03. Configuração de sistemas

Requisitos

- *RF 18.* Coordenador ou Analista mantém fontes.
- *RF 19.* Coordenador ou Analista mantém módulos.
- *RF 20.* Coordenador ou Analista mantém sistemas.
- *RF 21.* Coordenador ou Analista mantém versões.

Restrições

- *Pré-condição* . Coordenador ou Analista logado no sistema (UC 01.01).
- *Pós-condição* . Sistema insere nova ligação.
- *Pós-condição* . Sistema insere, atualiza ou remove ligação.

Cenários

Configuração de sistemas {Principal}.
1) Coordenador ou Analista seleciona o sistema.
2) Coordenador ou Analista seleciona a versão.
3) Coordenador ou Analista seleciona o módulo.
4) Coordenador ou Analista seleciona o fonte.
5) Sistema insere ligação.

Sistema não cadastrado {Alternativo}.
1) No passo 1, se o sistema ainda não foi cadastrado, o Coordenador ou Analista deve cadastrar o novo sistema acessando o cadastro de sistemas.

Versão não cadastrada {Alternativo}.

1) No passo 2, se a versão ainda não foi cadastrada, o Coordenador ou Analista deve cadastrar a nova versão acessando o cadastro de versões.

Módulo não cadastrado {Alternativo}.
1) No passo 3, se o módulo ainda não foi cadastrado, o Coordenador ou Analista deve cadastrar o novo módulo acessando o cadastro de módulos.

Fonte não cadastrado {Alternativo}.
1) No passo 4, se o fonte ainda não foi cadastrado, o Coordenador ou Analista deve cadastrar o novo fonte acessando o cadastro de fontes.

UC 02.01. Avaliação da manutenção

Requisitos

- *RF 8.* Coordenador define prioridades para a manutenção.
- *RF 9.* Coordenador identifica tipo de manutenção.
- *RF 10.* Coordenador define um analista para a manutenção.

Restrições

- *Pré-condição* . Coordenador logado no sistema (UC 01.01).
- *Pré-condição* . Coordenador seleciona uma manutenção pendente com status de "Avaliação".
- *Pós-condição* . Sistema registra status da manutenção como "Análise e Problema".

Cenários

Avaliação da manutenção {Principal}.
1) Sistema apresenta detalhes da manutenção.
2) Coordenador informa provável módulo afetado.
3) Coordenador informa provável fonte afetado.
4) Coordenador escolhe um Analista para a manutenção (obrigatório).
5) Coordenador identifica o tipo de manutenção (obrigatório).
6) Coordenador seleciona uma prioridade para a manutenção (obrigatório).
7) Sistema registra status da manutenção como "Análise e Problema".

UC 02.02. Cadastro de clientes

Restrições

- *Pré-condição* . Coordenador logado no sistema (UC 01.01).
- *Pós-condição* . Sistema atualiza, insere ou remove cliente.

Cenários

Cadastro de funções {Principal}.
1) Sistema apresenta uma lista de clientes cadastrados.
2) Coordenador escolhe um item da lista, acessando o link "Editar".
3) Coordenador informa o nome do cliente (obrigatório).
4) Coordenador informa o nome fantasia do cliente.
5) Coordenador informa a razão social do cliente.
6) Coordenador informa CNPJ do cliente (obrigatório).
7) Coordenador informa e-mail do cliente (obrigatório).
8) Coordenador informa fone do cliente (obrigatório).
9) Coordenador informa o país do cliente.
10) Coordenador informa o estado do cliente.
11) Coordenador informa a cidade do cliente.
12) Coordenador informa o bairro do cliente.
13) Coordenador informa o endereço do cliente.

14) Coordenador informa o CEP do cliente (obrigatório).
15) Coordenador identifica sistemas e versões do cliente (obrigatório).
16) Sistema atualiza cliente.

Ao acessar o link "Cadastrar um novo" {Alternativo}.
1) No passo 2, o Coordenador pode acessar o link "Cadastrar um novo", fazendo com que o sistema retome o fluxo no passo 3.

Ao acessar o link "Excluir" {Alternativo}.
1) No passo 2, o Coordenador pode acessar o link "Excluir", fazendo com que o sistema remova o cliente.

UC 02.03. Cadastro de funções

Restrições

- *Pré-condição* . Coordenador logado no sistema (UC 01.01).
- *Pós-condição* . Sistema atualiza função.

Cenários

Cadastro de funções {Principal}.
1) Sistema apresenta uma lista de funções cadastradas.
2) Coordenador escolhe um item da lista, acessando o link "Editar".
3) Sistema entra na tela de cadastro de função.
4) Coordenador preenche a descrição da função (obrigatório).
5) Sistema atualiza função.

UC 02.04. Cadastro de prioridades

Restrições

- *Pré-condição* . Coordenador logado no sistema (UC 01.01).
- *Pós-condição* . Sistema atualiza, insere ou remove prioridade.

Cenários

Cadastro de prioridades {Principal}.
1) Sistema apresenta uma lista de prioridades cadastradas.
2) Coordenador escolhe um item da lista, acessando o link "Editar".
3) Sistema entra na tela de cadastro de prioridade.
4) Coordenador preenche a descrição da prioridade (obrigatório).
5) Sistema atualiza prioridade.

Ao acessar o link "Cadastrar um novo" {Alternativo}.
1) No passo 2, o Coordenador pode acessar o link "Cadastrar um novo", fazendo com que o sistema retome o fluxo no passo 3.

Ao acessar o link "Excluir" {Alternativo}.
1) No passo 2, o Coordenador pode acessar o link "Excluir", fazendo com que o sistema remova a prioridade.

UC 02.05. Cadastro de status da manutenção

Restrições

- *Pré-condição* . Coordenador logado no sistema (UC 01.01).

- *Pós-condição* . Sistema atualiza status.

Cenários

Cadastro de status da manutenção {Principal}.
1) Sistema apresenta uma lista de status cadastrados.
2) Coordenador escolhe um item da lista, acessando o link "Editar".
3) Sistema entra na tela de cadastro de status.
4) Coordenador preenche a descrição do status (obrigatório).

UC 02.06. Cadastro de tipos de manutenção

Restrições

- *Pré-condição* . Coordenador logado no sistema (UC 01.01).
- *Pós-condição* . Sistema atualiza, insere ou remove tipo.

Cenários

Cadastro de tipos de manutenção {Principal}.
1) Sistema apresenta uma lista de tipos cadastrados.
2) Coordenador escolhe um item da lista, acessando o link "Editar".
3) Sistema entra na tela de cadastro de tipos.
4) Coordenador preenche a descrição do tipo (obrigatório).
5) Sistema atualiza tipo.

Ao acessar o link "Cadastrar um novo" {Alternativo}.
1) No passo 2, o Coordenador pode acessar o link "Cadastrar um novo", fazendo com que o sistema retome o fluxo no passo 3.

Ao acessar o link "Excluir" {Alternativo}.
1) No passo 2, o Coordenador pode acessar o link "Excluir", fazendo com que o sistema remova o tipo.

UC 02.07. Cadastro de usuários

Restrições

- *Pré-condição* . Coordenador logado no sistema (UC 01.01).
- *Pós-condição* . Sistema atualiza, insere ou remove usuário.

Cenários

Cadastro de funções {Principal}.
1) Sistema apresenta uma lista de usuários cadastrados.
2) Coordenador escolhe um item da lista, acessando o link "Editar".
3) Coordenador escolhe a empresa do usuário (obrigatório).
4) Coordenador informa o nome do usuário (obrigatório).
5) Coordenador escolhe a função do usuário (obrigatório).
6) Coordenador informa e-mail do usuário (obrigatório).
7) Coordenador informa senha do usuário (obrigatório).
8) Coordenador define o valor da hora do usuário (obrigatório).
9) Sistema atualiza usuário.

Ao acessar o link "Cadastrar um novo" {Alternativo}.
1) No passo 2, o Coordenador pode acessar o link "Cadastrar um novo", fazendo com que o sistema retome o fluxo no passo 3.

Ao acessar o link "Excluir" {Alternativo}.
1) No passo 2, o Coordenador pode acessar o link "Excluir", fazendo com que o sistema remova o usuário.

UC 03.01. Análise e problema da manutenção

Requisitos

- *RF 11.* Analista define a estimativa de custo para a manutenção.
- *RF 12.* Analista define a estimativa de tempo para a manutenção.
- *RF 13.* Analista anexa os documentos modificados na manutenção.
- *RF 14.* Analista define um programador para a manutenção.

Restrições

- *Pré-condição* . Analista logado no sistema (UC 01.01).
- *Pré-condição* . Analista seleciona uma manutenção pendente com status de "Análise e Problema".
- *Pós-condição* . Sistema registra status da manutenção como "Aprovação" ou "Implementação".

Cenários

Análise e problema da manutenção {Principal}.
1) Sistema mostra detalhes da manutenção.
2) Analista define estimativa de custo.
3) Analista define estimativa de tempo.
4) Analista anexa os documentos modificados e descreve o que foi modificado em cada documento.
5) Analista escolhe um programador para a manutenção (obrigatório).
6) Analista encaminha manutenção para o solicitante.
7) Sistema registra status da manutenção como "Aprovação".

Encaminhar para programador {Alternativo}.
1) No passo 6, o Analista pode encaminhar a manutenção para um programador.
2) Sistema registra status da manutenção como "Implementação".

UC 03.02. Aceitação e revisão da manutenção

Requisitos

- *RF 15.* Analista aprova a lista de produtos modificados pela manutenção.

Restrições

- *Pré-condição* . Analista logado no sistema (UC 01.01).
- *Pré-condição* . Analista seleciona uma manutenção pendente com status de "Aceitação e Revisão".
- *Pós-condição* . Sistema registra status da manutenção como "Homologação", "Implementação" ou "Concluída".

Cenários

Aceitação e revisão da manutenção {Principal}.
1) Sistema apresenta detalhes da manutenção.
2) Sistema apresenta todos os produtos gerados durante a manutenção.
3) Analista verifica os produtos gerados e aprova item por item.
4) Analista encaminha manutenção para o solicitante.
5) Sistema registra status da manutenção como "Homologação".

Reprovar item {Alternativo}.
1) No passo 3, o Analista pode reprovar um item.
2) Analista não encaminha manutenção para o solicitante.
3) Analista encaminha manutenção para o programador.
4) Sistema registra status da manutenção como "Implementação".

Concluir manutenção {Alternativo}.

1) No passo 4, o Analista pode não encaminhar a manutenção para o solicitante.
2) Analista não encaminha manutenção para o programador.
3) Sistema registra status da manutenção como "Concluída".

UC 04.01. Implementação da manutenção

Requisitos

- *RF 16.* Programador anexa os fontes modificados na manutenção.
- *RF 17.* Programador anexa um instalador do software para o Solicitante.

Restrições

- *Pré-condição* . Programador logado no sistema (UC 01.01).
- *Pré-condição* . Programador seleciona uma manutenção pendente com status de "Implementação".
- *Pós-condição* . Sistema registra status da manutenção como "Aceitação e Revisão".

Cenários

Implementação da manutenção {Principal}.
1) Sistema mostra detalhes da manutenção.
2) Programador anexa os fontes modificados e descreve o que foi modificado em cada fonte.
3) Programador anexa instalador para solicitante.
4) Sistema registra status da manutenção como "Aceitação e Revisão".

UC 05.01. Solicitação de manutenção

Requisitos

- *RF 5.* Solicitante faz a solicitação de manutenção via WEB.

Restrições

- *Pré-condição* . Solicitante logado no sistema (UC 01.01).
- *Pós-condição* . Sistema registra status da manutenção como "Avaliação".

Cenários

Solicitar manutenção {Principal}.
1) Solicitante entra na tela de solicitação de manutenção.
2) Sistema identifica sistemas e versões da empresa do solicitante.
3) Solicitante escolhe o sistema e a versão (obrigatório).
4) Solicitante descreve a manutenção (obrigatório).
5) Solicitante anexa uma imagem ou esboço da tela.
6) Sistema registra status da manutenção como "Avaliação".

UC 05.02. Aprovação da manutenção

Requisitos

- *RF 6.* Solicitante aprova orçamento da manutenção.

Restrições

- *Pré-condição* . Solicitante logado no sistema (UC 01.01).
- *Pré-condição* . Solicitante seleciona uma manutenção pendente com o status de "Aprovação".
- *Pós-condição* . Sistema registra status da manutenção como "Análise e Problema".

Cenários

Aprovar orçamento da manutenção {Principal}.
1) Sistema mostra detalhes do pedido.
2) Solicitante assinala a aprovação do orçamento.

3) Sistema registra status da manutenção como "Análise e Problema".

Reprovar orçamento {Alternativo}.
1) No passo 2, se o orçamento for reprovado, o solicitante deve justificar o por quê desta reprovação.

UC 05.03. Homologação da manutenção

Requisitos

- *RF 7.* Solicitante homologa manutenção.

Restrições

- *Pré-condição* . Solicitante logado no sistema (UC 01.01).
- *Pré-condição* . Solicitante seleciona uma manutenção pendente com o status de "Homologação".
- *Pós-condição* . Sistema registra status da manutenção como "Concluída" ou “Análise e Problema”.

Cenários

Homologar manutenção {Principal}.
1) Sistema mostra detalhes do pedido.
2) Solicitante assinala a homologação da manutenção.
3) Sistema registra status da manutenção como “Concluída”.

Não homologar {Alternativo}.

1) No passo 2, se a homologação não for assinalada, o solicitante deve justificar o por quê desta não homologação.

2) Sistema registra status da manutenção como “Análise e Problema”.

APÊNDICE B – Dicionário de dados

ARQ_HIS

Armazena os documentos pela ordem de serviço.

Nome	*Código*	*Anotação*	*Tipo de dado*	*Chave*
cod_sta	COD_STA	Código do status	INTEGER	TRUE
cod_os	COD_OS	Código da ordem de serviço	INTEGER	TRUE
cod_arq	COD_ARQ	Código do documento	INTEGER	TRUE
desc_arq_his	DESC_ARQ_HIS	Descrição do documento modificado	VARCHAR(255)	FALSE
local_arq_his	LOCAL_ARQ_HIS	Local do documento modificado	VARCHAR(255)	FALSE
aval_arq_his	AVAL_ARQ_HIS	Aval do documento modificado	SMALLINT	FALSE

ARQUIVO

Armazena os documentos

Nome	*Código*	*Anotação*	*Tipo de dado*	*Chave*
cod_arq	COD_ARQ	Código do documento	INTEGER	TRUE
nome_arq	NOME_ARQ	Nome do documento	VARCHAR(50)	FALSE

CLI_VER

Armazena os sistemas e versões do cliente.

Nome	*Código*	*Anotação*	*Tipo de dado*	*Chave*
cod_cli	COD_CLI	Código do cliente	INTEGER	TRUE
cod_sis	COD_SIS	Código do sistema	INTEGER	TRUE
cod_ver	COD_VER	Código da versão	INTEGER	TRUE

CLIENTE

Armazena as informações do cliente.

Nome	*Código*	*Anotação*	*Tipo de dado*	*Chave*
cod_cli	COD_CLI	Código do cliente	INTEGER	TRUE
cod_uf	COD_UF	Código do estado	INTEGER	FALSE
nome_cli	NOME_CLI	Nome do cliente	VARCHAR(50)	FALSE
nmfantasia_cli	NMFANTASIA_CLI	Nome fantasia do cliente	VARCHAR(50)	FALSE
nmrazaosocial_cli	NMRAZAOSOCIAL_CLI	Razão social do cliente	VARCHAR(100)	FALSE
email_cli	EMAIL_CLI	E-mail do cliente	VARCHAR(50)	FALSE
cnpj_cli	CNPJ_CLI	CNPJ do cliente	VARCHAR(30)	FALSE
pais_cli	PAIS_CLI	País do cliente	VARCHAR(50)	FALSE
cidade_cli	CIDADE_CLI	Cidade do cliente	VARCHAR(50)	FALSE
bairro_cli	BAIRRO_CLI	Bairro do cliente	VARCHAR(50)	FALSE
rua_cli	RUA_CLI	Rua do cliente	VARCHAR(50)	FALSE
cep_cli	CEP_CLI	CEP do cliente	VARCHAR(10)	FALSE
fone_cli	FONE_CLI	Fone do cliente	VARCHAR(20)	FALSE

ESTADO

Armazena os estados brasileiros.

Nome	*Código*	*Anotação*	*Tipo de dado*	*Chave*
cod_uf	COD_UF	Código do estado	INTEGER	TRUE
nome_uf	NOME_UF	Nome do estado	VARCHAR(50)	FALSE
sigla_uf	SIGLA_UF	Sigla do estado	VARCHAR(2)	FALSE

FON_HIS

Armazena os fontes gerados pela ordem de serviço.

Nome	*Código*	*Anotação*	*Tipo de dado*	*Chave*
cod_sta	COD_STA	Código do status	INTEGER	TRUE
cod_os	COD_OS	Código da ordem de serviço	INTEGER	TRUE
cod_fon	COD_FON	Código do fonte	INTEGER	TRUE
desc_fon_his	DESC_FON_HIS	Descrição do fonte modificado	VARCHAR(255)	FALSE
local_fon_his	LOCAL_FON_HIS	Local do fonte modificado	VARCHAR(255)	FALSE
aval_fon_his	AVAL_FON_HIS	Aval do fonte modificado	SMALLINT	FALSE

FONTE

Armazena os fontes.

Nome	*Código*	*Anotação*	*Tipo de dado*	*Chave*
cod_fon	COD_FON	Código do fonte	INTEGER	TRUE
nome_fon	NOME_FON	Nome do fonte	VARCHAR(30)	FALSE

FUNÇÃO

Armazena as funções ou perfil do usuário.

Nome	*Código*	*Anotação*	*Tipo de dado*	*Chave*
cod_fun	COD_FUN	Código da função	INTEGER	TRUE
desc_fun	DESC_FUN	Descrição da função	VARCHAR(30)	FALSE

HISTÓRICO

Armazena o histórico da ordem de serviço.

Nome	*Código*	*Anotação*	*Tipo de dado*	*Chave*
cod_sta	COD_STA	Código do status	INTEGER	TRUE
cod_os	COD_OS	Código da ordem de serviço	INTEGER	TRUE
cod_usu	COD_USU	Código do usuário	INTEGER	FALSE
dtinicio_his	DTINICIO_HIS	Data inicial do histórico	VARCHAR(10)	FALSE
dtfim_his	DTFIM_HIS	Data final do histórico	VARCHAR(10)	FALSE
dttroca_his	DTTROCA_HIS	Data de encaminhamento do histórico	VARCHAR(10)	FALSE
temporeal_his	TEMPOREAL_HIS	Duração real do histórico	INTEGER	FALSE
custoreal_his	CUSTOREAL_HIS	Custo real do histórico	INTEGER	FALSE

dsgeral_his	DSGERAL_HIS	Descrição geral do histórico	VARCHAR(255)	FALSE
aval_his	AVAL_HIS	Aval do histórico	SMALLINT	FALSE

MÓDULO

Armazena os módulos.

Nome	*Código*	*Anotação*	*Tipo de dado*	*Chave*
cod_mod	COD_MOD	Código do módulo	INTEGER	TRUE
nome_mod	NOME_MOD	Nome do módulo	VARCHAR(50)	FALSE

ORDEM SERVIÇO

Armazena as informações da ordem de serviço.

Nome	*Código*	*Anotação*	*Tipo de dado*	*Chave*
cod_os	COD_OS	Código da ordem de serviço	INTEGER	TRUE
cod_sta	COD_STA	Código do status	INTEGER	FALSE
cod_usu	COD_USU	Código do usuário	INTEGER	FALSE
cod_tip	COD_TIP	Código do tipo de manutenção	INTEGER	FALSE
cod_mod	COD_MOD	Código do módulo	INTEGER	FALSE
cod_pri	COD_PRI	Código da prioridade	INTEGER	FALSE
cod_fon	COD_FON	Código do fonte	INTEGER	FALSE
cod_sis	COD_SIS	Código do sistema	INTEGER	FALSE
cod_cli	COD_CLI	Código do cliente	INTEGER	FALSE
cod_ver	COD_VER	Código da versão	INTEGER	FALSE
data_os	DATA_OS	Data de abertura da ordem de serviço	VARCHAR(10)	FALSE
dsgeral_os	DSGERAL_OS	Descrição geral de ordem de serviço	VARCHAR(255)	FALSE
localanexo_os	LOCALANEXO_OS	Local de anexos da ordem de serviço	VARCHAR(255)	FALSE
localprog_os	LOCALPROG_OS	Local do programa final resultante da ordem de serviço	VARCHAR(255)	FALSE
ecusto_os	ECUSTO_OS	Estimativa de custo para desenvolvimento da ordem de serviço	INTEGER	FALSE
etempo_os	ETEMPO_OS	Estimativa de tempo para desenvolvimento da ordem de serviço	INTEGER	FALSE

PRIORIDADE

Armazena as prioridades da ordem de serviço.

Nome	*Código*	*Anotação*	*Tipo de dado*	*Chave*
cod_pri	COD_PRI	Código da prioridade	INTEGER	TRUE
desc_pri	DESC_PRI	Descrição da prioridade	VARCHAR(10)	FALSE

SISTEMA

Armazena os sistemas.

Nome	*Código*	*Anotação*	*Tipo de dado*	*Chave*
cod_sis	COD_SIS	Código do sistema	INTEGER	TRUE
desc_sis	DESC_SIS	Descrição do sistema	VARCHAR(50)	FALSE

STATUS

Armazena as etapas correspondentes ao processo de manutenção.

Nome	*Código*	*Anotação*	*Tipo de dado*	*Chave*
cod_sta	COD_STA	Código do status	INTEGER	TRUE
desc_sta	DESC_STA	Descrição do status	VARCHAR(30)	FALSE

TIPO

Armazena o tipo de manutenção.

Nome	*Código*	*Anotação*	*Tipo de dado*	*Chave*
cod_tip	COD_TIP	Código do tipo de manutenção	INTEGER	TRUE
desc_tip	DESC_TIP	Descrição do tipo de manutenção	VARCHAR(30)	FALSE

USUÁRIO

Armazena as informações do usuário.

Nome	*Código*	*Anotação*	*Tipo de dado*	*Chave*
cod_usu	COD_USU	Código do usuário	INTEGER	TRUE
cod_cli	COD_CLI	Código do cliente	INTEGER	FALSE
cod_fun	COD_FUN	Código da função	INTEGER	FALSE
nome_usu	NOME_USU	Nome do usuário	VARCHAR(50)	FALSE
email_usu	EMAIL_USU	E-mail do usuário	VARCHAR(50)	FALSE
senha_usu	SENHA_USU	Senha do usuário	VARCHAR(10)	FALSE
vlrhora_usu	VLRHORA_USU	Valor cobrado por hora de trabalho do usuário	INTEGER	FALSE

VER_ARQ

Armazena os documentos do sistema e versão.

Nome	*Código*	*Anotação*	*Tipo de dado*	*Chave*
cod_sis	COD_SIS	Código do sistema	INTEGER	TRUE
cod_ver	COD_VER	Código da versão	INTEGER	TRUE
cod_arq	COD_ARQ	Código do documento	INTEGER	TRUE
local_ver_arq	LOCAL_VER_ARQ	Local do documento	VARCHAR(255)	FALSE

VER_MOD

Faz a ligação sistema, versão, módulo e fonte.

Nome	*Código*	*Anotação*	*Tipo de dado*	*Chave*
cod_sis	COD_SIS	Código do sistema	INTEGER	TRUE
cod_ver	COD_VER	Código da versão	INTEGER	TRUE
cod_mod	COD_MOD	Código do módulo	INTEGER	TRUE
cod_fon	COD_FON	Código do fonte	INTEGER	TRUE
local_ver_mod	LOCAL_VER_MOD	Local do produto original	VARCHAR(255)	FALSE

VERSÃO

Armazena as versões do sistema.

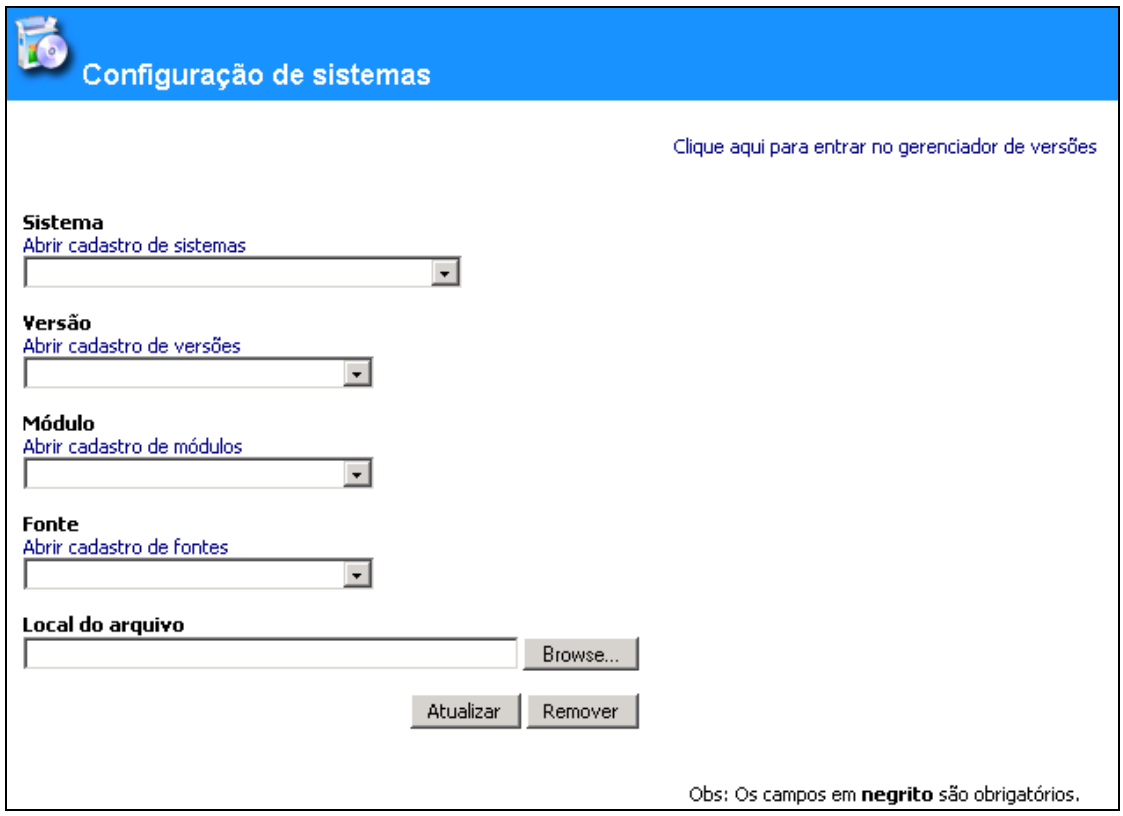
Nome	*Código*	*Anotação*	*Tipo de dado*	*Chave*
cod_sis	COD_SIS	Código do sistema	INTEGER	TRUE
cod_ver	COD_VER	Código da versão	INTEGER	TRUE
nome_ver	NOME_VER	Nome da versão	VARCHAR(10)	FALSE

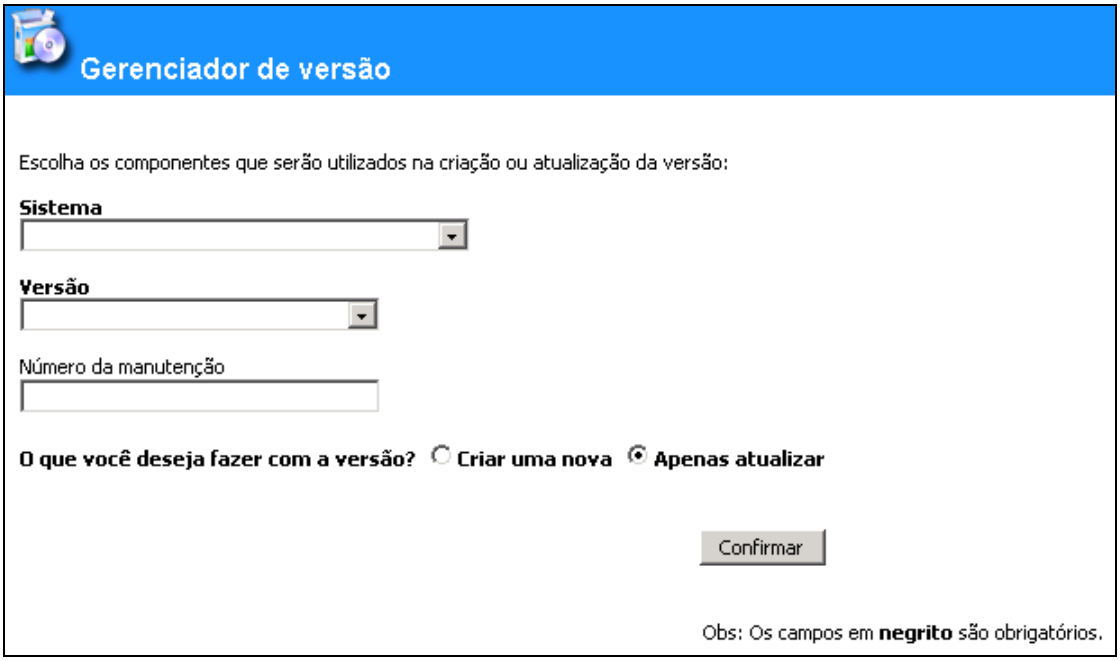
APÊNDICE C – Funcionalidades complementares

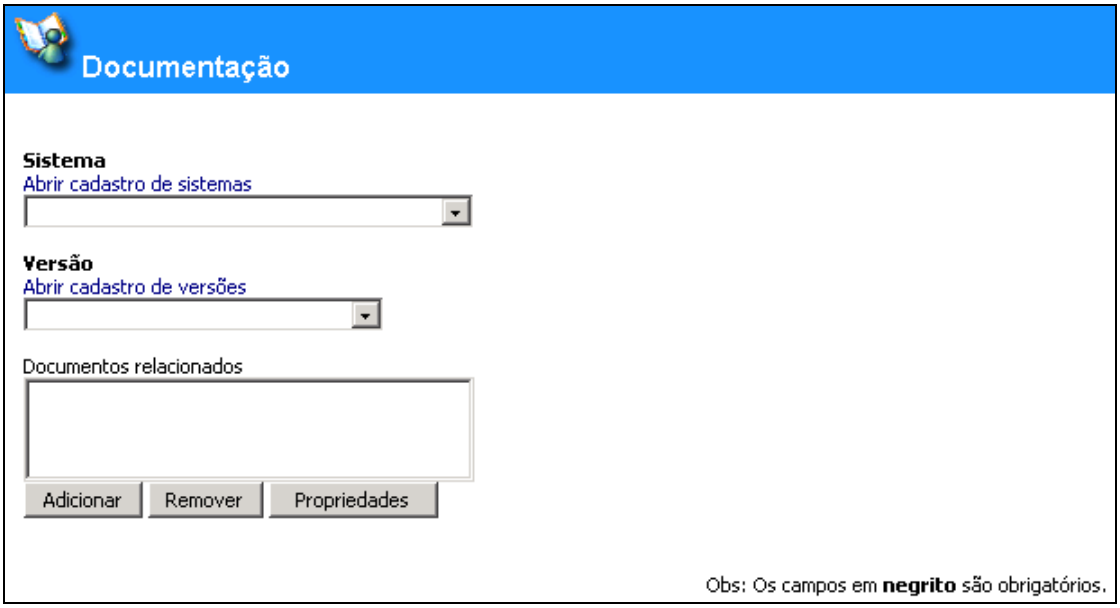
Este Apêndice apresenta as telas referentes às funcionalidades de suporte à execução do *workflow*. Visando facilitar o entendimento e sua utilização as telas estão associadas aos casos de uso identificados neste trabalho.

UC 02.02. Cadastro de clientes

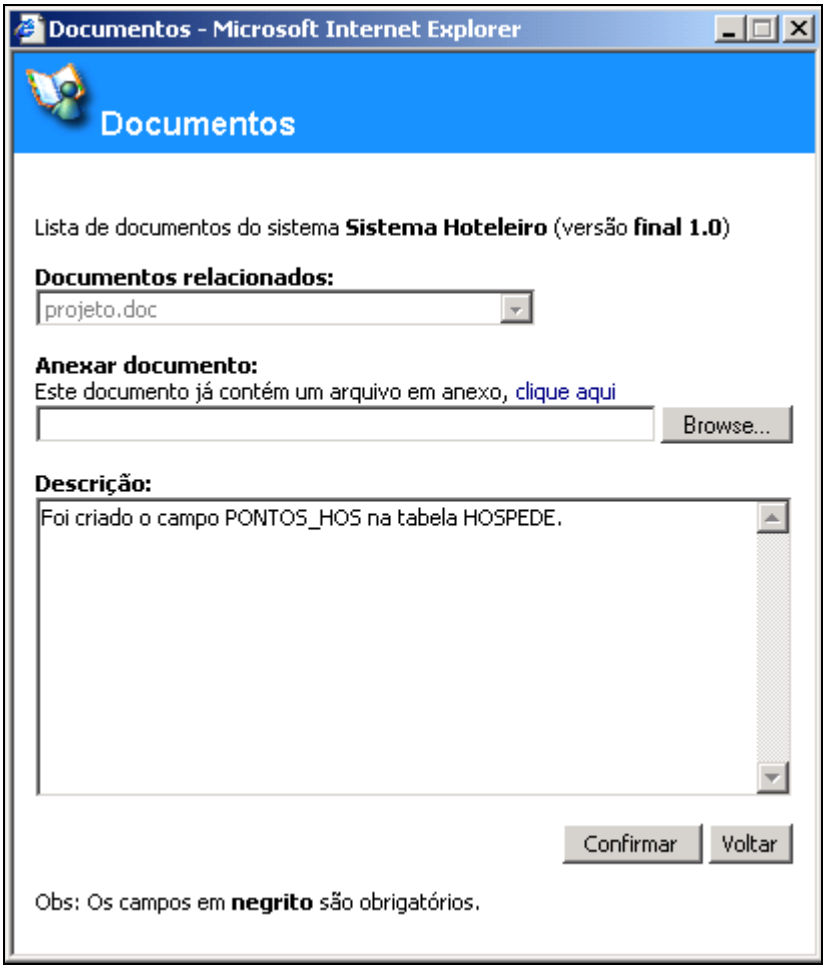
UC 01.03. Configuração de sistemas




Gerenciador de versão
Escolha os componentes que serão utilizados na criação ou atualização da versão:
Sistema
Versão
Número da manutenção
O que você deseja fazer com a versão?
Criar uma nova
Apenas atualizar
Confirmar
Obs: Os campos em negrito são obrigatórios.

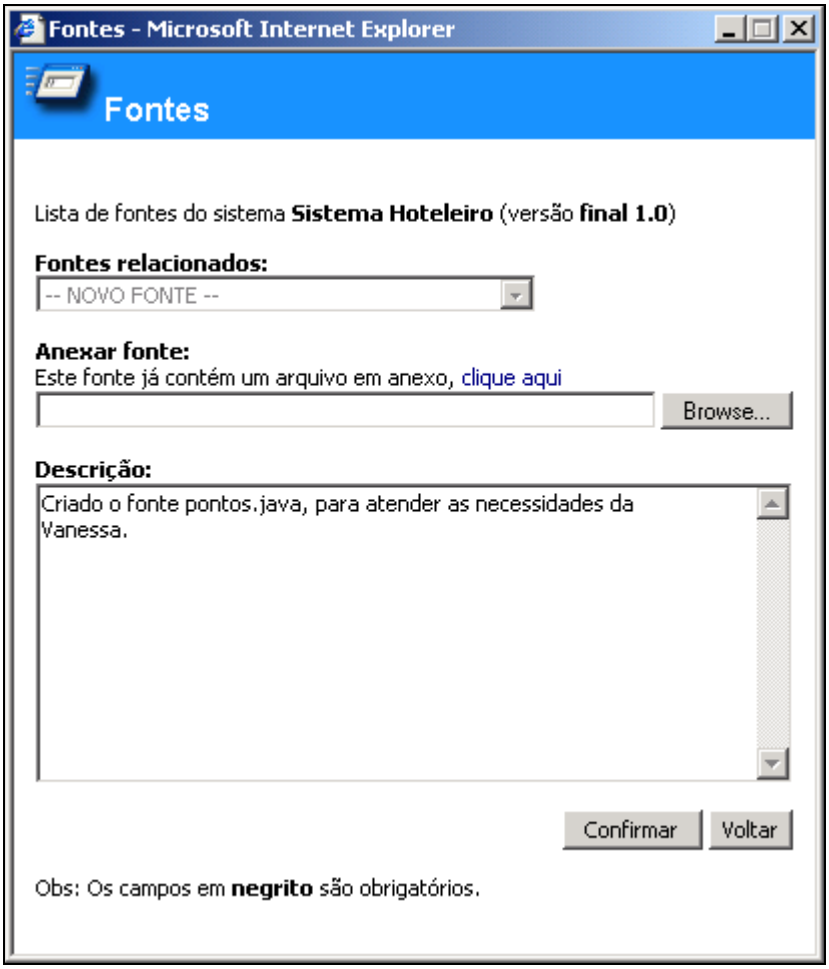

Documentação
Sistema
Abrir cadastro de sistemas
Versão
Abrir cadastro de versões
Documentos relacionados
Adicionar
Remover
Propriedades
Obs: Os campos em negrito são obrigatórios.

UC 03.01. Análise e problema da manutenção

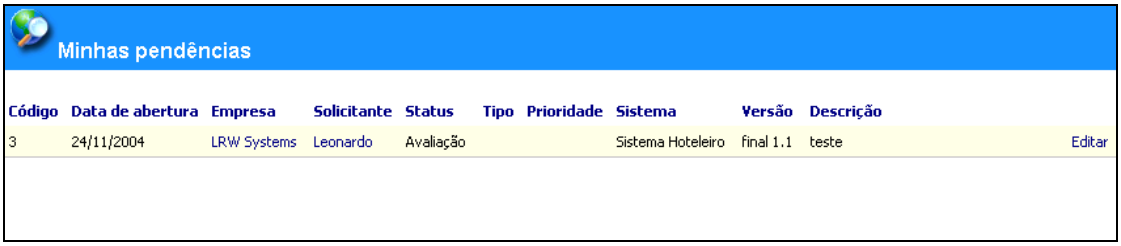


UC 02.03. Cadastro de funções

UC 04.01. Implementação da manutenção

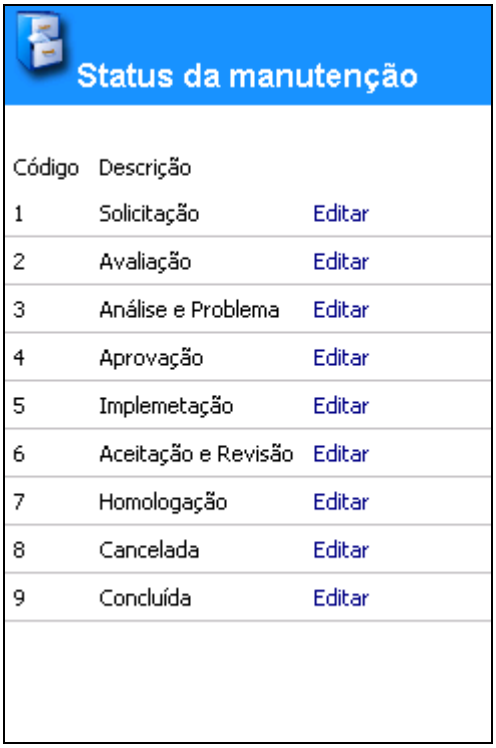


UC 01.02. Consultar manutenções



UC 02.04. Cadastro de prioridades

UC 02.05. Cadastro de status da manutenção



UC 02.06. Cadastro de tipos de manutenção

UC 02.07. Cadastro de usuários