PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS

Programa de Pós-graduação em Engenharia Elétrica

MODELO DE PROCESSO DE TESTES PARA SISTEMAS DE SOFTWARE CRÍTICOS

Maria Jose Silva Mesquita Cardoso

Belo Horizonte

Março de 2010

Maria José Silva Mesquita Cardoso

MODELO DE PROCESSO DE TESTES PARA SISTEMAS DE SOFTWARE CRÍTICOS

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica, da Pontifícia Universidade Católica de Minas Gerais, como requisito parcial para obtenção do grau de Mestre em Engenharia Elétrica.

Linha de Pesquisa: Teste de Sistemas Críticos.

Orientador: Prof. Dr. Carlos Alberto Marques Pietrobon.

Março de 2010

Belo Horizonte

FICHA CATALOGRÁFICA
Elaborada pela Biblioteca da Pontifícia Universidade Católica de Minas Gerais

Cardoso, Maria José Silva Mesquita
C268m Modelo de processo de testes para sistemas de software críticos / Maria José Silva Mesquita Cardoso. Belo Horizonte, 2010.
170.f.

Orientador: Carlos Alberto Marques Pietrobon
Dissertação (Mestrado) – Pontifícia Universidade Católica de Minas Gerais. Programa de Pós-Graduação em Engenharia Elétrica.

1. Engenharia de software. 2. Software -Testes. 3. Software de sistemas. I. Pietrobon, Carlos Alberto Marques. II. Pontifícia Universidade Católica de Minas Gerais. III. Título.

CDU: 681.3

Maria José Silva Mesquita Cardoso

MODELO DE PROCESSO DE TESTES PARA SISTEMAS DE SOFTWARE CRÍTICOS

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica, da Pontifícia Universidade Católica de Minas Gerais, como requisito parcial para obtenção do grau de Mestre em Engenharia Elétrica.
Linha de Pesquisa: Teste de Sistemas Críticos.

_____

Orientador: Prof. Dr. Carlos Alberto Marques Pietrobon

_____

Prof. Dr. Alexei Manso Corrêa Machado
Pontifícia Universidade Católica de Minas Gerais
PPGEE

_____

Prof. Dr. Ricardo de Oliveira Duarte
Universidade Federal de Minas Gerais
Departamento de Engenharia Eletrônica

Belo Horizonte, 12 de março de 2010

A Deus, por seu infinito amor e cuidado para comigo, por me capacitar e me ajudar a vencer mais esta etapa da minha vida.

A minha mãe, minha fortaleza.

AGRADECIMENTOS

A Deus, por seu amor infinito, por me amparar e fazer-me enxergar a motivação para a minha vida, por me conduzir pelos caminhos que trilhei e permitir que eu vencesse mais este desafio em minha vida. A minha mãe, por seu amor incondicional, por seu exemplo de luta e persistência. A meu pai, por seu exemplo de honestidade e seu cuidado com a execução de tudo da forma mais correta possível. Mesmo não estando mais presente, sempre foi inspiração constante em meu caminho.

A meu marido, Julio, por sua compreensão e apoio nos momentos de dificuldade. A meus filhos por me incentivarem a estudar, me apoiarem a cada vitória, pelo carinho e compreensão a mim demonstrados e por entenderem minha ausência.

A meu orientador, Professor Pietro, pela orientação, incentivo e paciência ao longo desses anos, por acreditar em mim, pelo aprendizado que me proporcionou muitas vezes, alterando seu horário de trabalho para adequar-se ao meu.

Aos colegas de mestrado Ana Paula, Leia Assis, Josy, Ana Flavia, Camila, André, Jair Maquea, Isabela, e a todos os outros, pela amizade, alegria contagiante, oportunidade de conviver com esta juventude em ebulição e pela acolhida carinhosa e inesquecível. Aos funcionários do PPGEE pela atenção e disponibilidade para resolver todos os problemas e o apoio durante o curso.

Ao Banco Itaú pelo apoio financeiro e pelo incentivo fornecido no inicio do mestrado, sem o qual teria sido impossível iniciar. A meus chefes, José Brasil Lessa Correa e Gustavo Ladeira Resende, pelo apoio, amizade e ensinamentos em anos de trabalho na área de Informática do Banco Itaú, agradeço, também, aos colegas de trabalho e convivência de duas décadas de amizade e crescimento profissional.

Ao pessoal da Rumo Informática, pela amizade, apoio, incentivo e oportunidade fornecida para colocar em prática os resultados dos estudos na área de testes de sistemas. Ao Banco Rural pela oportunidade proporcionada, meios para enfrentar desafios e apoio para solucioná-los. Aos amigos Anderson, Bothrel, Felipe, Igor, Isabel, Luciana, José Maurício, Marcelo Farinha e a todos os outros pela convivência diária, apoio, amizade, solidariedade e crescimento conjunto.

RESUMO

A sociedade do século XXI está cada vez mais dependente de sistemas e softwares que facilitam e viabilizam atividades que, até poucos anos atrás, não eram imaginadas possíveis. A telecomunicação, a automação dos eletrodomésticos, dos veículos, o transporte aéreo, metroviário, urbano, o controle da distribuição de energia, a segurança física das pessoas, a dependência da conexão e comunicação com pessoas fisicamente separadas e outros estão fazendo com que o homem fique mais e mais a mercê de uma tecnologia que deve funcionar com precisão e por todo o tempo. A menos de 5 anos, demoraríamos a enumerar alguns sistemas críticos. Atualmente é difícil relacioná-los em um único parágrafo. Nesse contexto, esta dissertação apresenta um Modelo de Processo de Testes para Sistemas Críticos, embasado no planejamento, geração e execução de testes vinculados aos perigos e riscos a que o sistema está exposto. No modelo proposto, a construção do processo de teste deve iniciar-se juntamente com o desenvolvimento do sistema. As equipes de teste, desenvolvimento e especialistas do domínio interagem nos módulos de planejamento e geração do processo de testes. Inicialmente é feito um levantamento dos perigos e riscos dos sistemas sendo identificados Fatores de Criticidade (criticality), em função deles. Os processos, para obter a sua mitigação, são definidos em uma atividade prévia, a Análise de Segurança do Sistema. Nessa, são definidos os homologadores responsáveis pela conferência e liberação dos artefatos testados. Quando da geração dos testes, são construídos os casos de testes específicos para os artefatos responsáveis pelo controle desses perigos, o que implica em maior quantidade de testes e testes mais robustos para perigos de maior criticidade. Esses artefatos são classificados como artefatos de missão crítica e submetidos a testes mais robustos e controlados. A execução de testes previamente planejados e construídos permite agilidade e rápida verificação da existência de erros e liberação dos artefatos corretos com mais segurança. Em se tratando de sistemas críticos, esse é um requisito essencial. Foi construída uma ferramenta o Sistema de Apoio aos Testes de Sistemas Críticos (SATS), em que o conhecimento construído, é armazenado e disponibilizado para a equipe de testes. Os casos de testes cadastrados com os dados de entrada e resultados esperados conforme as bases de testes foram preenchidas. Por fim, a pesquisa apresenta um estudo de caso conduzido em uma instituição financeira de Belo Horizonte. As conclusões apresentadas satisfazem o objetivo de favorecer o planejamento, a construção do projeto e a execução dos testes, além de fornecer, aos desenvolvedores e homologadores, maior segurança para liberar os artefatos construídos e alterados. Essa é uma abordagem diferenciada para a melhoria do processo de teste. Este trabalho contribui para o desenvolvimento e consolidação do Projeto Discovery, que vem sendo desenvolvido por um grupo de pesquisadores do PPGEE - da Pontifícia Universidade Católica de Minas Gerais (PUCMINAS) e tem como perspectiva atuar na geração, visualização, preservação e descoberta do conhecimento.

Palavras-chave: Sistemas críticos, Fatores de Criticidade, análise de segurança crítica, modelos de melhoria de teste, MPT.BR, MPS.BR, projeto de teste, execução de testes, casos de teste, homologação de teste

ABSTRACT

The XXI century society is more dependent of systems and software that facilitate and make possible activities that, until few years ago, were not imagined possible. The telecommunication, the automation of household-electric, the vehicles, the air, subway and urban transportation, the control of the energy distribution, the physical security of the people, the dependence of the connection and communication with physically separated people and others are making the man more and more at the grace of a technology that must function precisely for all the time. Less than 5 years, we would delay to enumerate some critical systems. Currently it is difficult to relate them in one paragraph. In this context, this study presents a Critical Systems Test Process Model, based in planning, generation and execution of entailed tests to the hazards and risks to which the system is exposed. In the considered model, the construction of the test process must be initiated together with the development of the system. The test teams, the development team and the domain specialists interact in the modules of planning and generation of the test process. Initially a searching of the hazards and risks of the systems is made, identifying the Critical Factors (criticality). The processes, in order to get them working, are defined in a previous activity, the System Security Analysis. In this, the responsible approvers for the conference and release of the tested devices are defined. As for the generation of the tests, the cases of specific tests for the devices responsible for the control of these hazards are constructed, what implies in a bigger amount of tests and more robust tests for bigger criticality hazards. These devices are classified as devices of critical mission and they are submitted to even robust and controlled the tests. The execution of tests previously planned and constructed allows agility and speed in verification of the existence of errors and release of the correct safer devices. As for critical systems, this is an essential requirement. A SATS tool was constructed - System of Support to the Tests of Critical Systems - where the constructed knowledge, is stored and available for the test team. The test cases with input data and awaited results as the bases of tests had been filled. Finally, the research presents a case study lead in a financial institution of Belo Horizonte. The presented conclusions satisfy the objective of favoring the planning, the project construction and the tests execution, and also it supplies, the developers and approvers, with greater security to liberate the constructed and modified devices. This is an approach differentiated for the improvement of the test process. This work contributes for the development and consolidation of the Discovery Project, that’s being developed by a group of researchers of the Post-Graduation Program in Electric Engineering of the PUCMINAS and has as perspective to act in the generation, visualization, preservation and discovery of the knowledge.

Keywords: Critical Systems, Critical Factors, Critical Security Analysis, Test Improvement Models, MPT.BR, MPS.BR, Test Project, Test Execution, Test Cases, Test Approvals

LISTA DE FIGURAS

LISTA DE GRÁFICOS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CMM – Capability Maturity Model

CMMI – Capability Maturity Model Integration

COTS – *Commercial Off-The-Shelf Systems*

ETM – Equipe Técnica do Modelo

FCC – Fórum de Credenciamento e Controle

GPR – Gerência de Projetos

GRI – Gerência de Risco

MA – Modelo de Avaliação

MCT – Ministério de Ciência e Tecnologia

MN – Modelo de Negócios

MNC – Modelo de Negocio Cooperado

MPS-BR – Melhoria de Processo do Software Brasileiro

MPT.BR – Melhoria de Processo de Teste Brasileiro

MR – Modelo de Referência

MT – Modelo de Teste

PPGEE – Pós-Graduação de Engenharia Elétrica

PUCMINAS – Pontifícia Universidade Católica de Minas Gerais

SATS – Sistema de Apoio aos Testes de Sistemas Críticos

SBC – Sociedade Brasileira de Computação

SGBD – Sistema Gerenciador de Banco de Dados

SOFTEX – Associação para Promoção da Excelência do Software Brasileiro

TMM – *Test Maturity Model*

TPI – *Test Process Improvement*

VAL – Validação

VER – Verificação

SUMÁRIO

1 INTRODUÇÃO

Neste primeiro capítulo, situa-se o problema no item 1.1, mostrando a vulnerabilidade dos sistemas críticos. No item 1.2, relacionam-se os objetivos principais e secundários do trabalho. No item 1.3 e 1.4 explicita-se a relevância do tema e a motivação de sua escolha, finaliza-se com os itens 1.5 e 1.6, contendo o detalhamento do escopo da dissertação e a metodologia utilizada para sua confecção.

1.1 O Problema

Hoje em dia, os computadores estão cada vez mais sendo utilizados para controlar sistemas que tem de ser seguros e confiáveis. Confiabilidade e segurança são características que são exigidas implicitamente ou explicitamente em quase todos os sistemas, principalmente nas aplicações críticas (RODRIGUES, 2002).

As aplicações críticas são aquelas em que uma falha pode provocar severas perdas materiais, humanas e financeiras. Por exemplo: Geração de Energia Nuclear, Processos Químicos, Aviação Comercial e Área Aeroespacial, Transporte Metro-Ferroviário, Equipamentos Médicos, Indústria em Geral, Sistemas de Informação, Sistemas Bancários de Pagamentos, Transferências de Valores, Prevenção à Corrupção e Lavagem de Dinheiro e Controles Automáticos de Compra e Venda de Ações em Tempo Real.

Todas são aplicações que não podem ter seu funcionamento prejudicado ou paralisado, não apenas por problemas de acidentes, mas pelos benefícios que proporcionam à coletividade, e estes não podem deixar de ser oferecidos. Cada vez mais, o principal componente dos Sistemas de Supervisão e Controle de Aplicações Críticas é composto por computadores e softwares, cuja utilização é, e deverá continuar sendo, crescente no controle desses sistemas.

Aliada à complexidade dos componentes de hardware, vem acoplada a não menos complexa tarefa de produção de software (ALMEIDA JUNIOR, 2009). O processo natural de integração dos sistemas dos vários domínios faz com que,

mesmo os sistemas que não eram reconhecidos como críticos passem a ser assim classificados.

Podemos tratar a criticidade de duas formas: por meio de testes e por meio de processos. A utilização de testes é a abordagem tradicional para garantir a qualidade do software e ela ganha uma importância maior, quando se trata de uma aplicação crítica.

O processo de desenvolvimento de software utilizado para a construção do software é determinante na qualidade do software desenvolvido (CRUZ, CRESPO e ARGOLLO, 2006). Os modelos de melhoria e controle de qualidade de produção de software MPS.BR e CMMI são exemplos de propostas de processos e abordagens que consideram a atividade de testes, embora não detalhem como essa atividade deve ser executada, já o modelo MPT.BR (MPT, 2009), é uma proposta específica para processos de teste com qualidade.

O problema é definir o processo a ser utilizado para suportar o planejamento, controle e execução dos testes, apoiando as atividades de Verificação e Validação de sistemas críticos. Este trabalho é uma proposta de processo para testar sistemas críticos que considera o modelo MPS.BR.

1.2 Objetivos

1.2.1 Objetivo Principal

O objetivo principal deste trabalho é propor um Modelo de Processo de Testes específico para sistemas críticos, inserido no processo de desenvolvimento, instanciado a partir de um processo padrão da empresa. Esse modelo de Teste de Sistemas pretende incrementar a qualidade de softwares, tanto no momento de sua construção, quanto nos procedimentos de manutenção evolutiva ou corretiva. O modelo considerará o modelo de qualidade MPS-BR.

Essa proposta apresenta também o Sistema de Apoio a Testes de Sistemas Críticos (SATS), que é uma ferramenta para facilitar o planejamento e criação do processo de testes do sistema como um todo, controlar a execução dos testes de

cada artefato, controlar o registro de dados relativos a estes testes, controlar o registro do conhecimento construído relativo aos casos de uso, aos casos de testes e seus resultados esperados e acompanhar a situação dos testes a nível gerencial.

O trabalho aqui apresentado está alinhado com o quinto desafio proposto no seminário 'Desafios da pesquisa em computação 2006-2016' promovido pela Sociedade Brasileira de Computação (SBC), em 2006, que é: desenvolvimento tecnológico de qualidade, sistemas disponíveis, corretos, seguros, escaláveis persistentes e ubíquos, que foquem o bem-estar da sociedade (CARVALHO *et. al*, 2006).

Não é objetivo deste trabalho apresentar uma técnica ou método específico de teste, nem considerar um domínio específico da aplicação crítica ou uma abordagem para tratar aplicação crítica (por exemplo, sistemas tolerantes a falhas).

1.2.2 Objetivos Secundários

- Definir a adequação do MPS.BR para desenvolvimento de sistemas críticos;
- Incrementar o Projeto Discovery;
- Fornecer subsídios para empresas que queiram alcançar o nível ‘D’ do MPS.BR, ao cumprir os processos exigidos pelo MPS.BR nas atividades de Validação e Verificação; e
- Construir uma ferramenta para apoiar o modelo de processo. (SATS)

1.3 Relevância

A relevância desta pesquisa encontra-se na necessidade das organizações desenvolvedoras de verificar se os produtos e artefatos desenvolvidos para Sistemas Críticos estão sendo testados com a segurança que este tipo de sistema exige. Ao propor um modelo de teste para esses sistemas, caminha-se no sentido da busca à conformidade dos processos aos modelos de referência e maior segurança no acréscimo à garantia de qualidade desse tipo de sistema.

A relevância do trabalho também se deve à ferramenta que está sendo proposta para apoiar o planejamento, acompanhamento, controle, reuso e visualização dos testes de software, em seus vários níveis e momentos, durante o ciclo de desenvolvimento de um software. O modelo de testes proposto e a ferramenta auxiliam a área de qualidade da empresa na verificação do atendimento dos requisitos do Sistema e monitoramento dos Fatores de Criticidade do sistema.

1.4 Motivação

A principal motivação deste trabalho surgiu da carência das empresas em tratar softwares críticos, da experiência profissional da autora em desenvolvimento e testes de softwares e da pesquisa na literatura para solução de problemas relacionados a testes de softwares para Sistemas Críticos, (SAEED, LEMOS e ANDERSON, 1991; KUVAJA *et. al*, 1994; MOURA, 1996; STOREY, 1996; DIAS, 2000; ALMEIDA JUNIOR, 2003; MOLINARE, 2003; CRESPO, 2004), que apontam as seguintes carências:

- Falta de modelos de processos de testes definidos nas organizações;
- Testes inadequados de artefatos, rotinas e sistemas com consequente liberação de sistemas que não atendem aos requisitos explicitados na fase de projeto;
- Falta de planejamento, controle e acompanhamento do processo de testes dos Sistemas; e

- Falta de critérios e parâmetros claros para a definição dos Fatores de Criticidade e do Grau de criticidade dos sistemas.

Além disso, a utilização de software em substituição ao trabalho humano é um fato irreversível na sociedade moderna. Os controles de equipamentos e procedimentos de risco estão sendo transferidos para sistemas que, por si só, não tem poder de crítica e decisão, se não forem programados para tal.

Disponibilizar artefatos adequados e corretos é tarefa cada vez mais exigida. A democratização e fluidez da informação e dos serviços informatizados disponibilizam o correto e o incorreto com a mesma agilidade. Como consequência disso, a preocupação com a prevenção de erros e procedimentos inadequados é cada vez maior. Com isso, a definição de um modelo de testes baseado em normas de qualidade é de suma importância e relevância para as comunidades e traz grande interesse para seu desenvolvimento, ainda mais tratando-se de sistemas críticos.

1.5 Escopo

Este trabalho está inserido na Engenharia de Software, precisamente na área de processo de desenvolvimento de software com qualidade. Os principais temas a serem tratados são: sistemas críticos, fatores de criticidade dos sistemas, grau de criticidade, modelos de melhoria do processo de desenvolvimento de software, principalmente o MPS.BR, o CMMI e os modelos específicos de melhoria do processo de testes como o TPI, o TMM e o MPT.BR.

1.6 Metodologia

Conforme Vergara (2003), a metodologia utilizada neste trabalho pode ser dividida quanto aos fins e quanto aos meios. Quanto aos fins, a pesquisa é descritiva, uma vez que descreve um modelo de processo de testes a ser utilizado

em projetos de desenvolvimento de softwares, mais especificamente, nos sistemas críticos que, por sua própria característica de criticidade, exigem mais cuidado em seu desenvolvimento e liberação aos utilizadores.

Quanto aos meios, foi realizado um estudo bibliográfico, foram feitas pesquisas em livros, artigos de instituições, publicações especializadas, documentações técnicas, normas e/ou recomendações, monografias, dissertações e teses. Diversos sites da rede mundial de computadores foram consultados e a participação em fóruns de discussão sobre sistemas críticos, acidentes aéreos e problemas de operação relatados foi necessária. O modelo proposto levou em consideração o nível ‘D’ do MPS.BR, em que se localiza o processo de Verificação, que é o mais aderente ao processo de testes objeto deste trabalho.

Foi realizada uma pesquisa bibliográfica para o estudo das normas e padrões de qualidade e das normas que definem, tratam e regulamentam os sistemas críticos.

1.7 Organização da Dissertação

Esta dissertação está organizada em sete capítulos. O primeiro faz a introdução ao tema, expõe o problema e apresenta os objetivos principais e secundários do trabalho em linhas gerais.

O segundo capítulo define os Sistemas Críticos, descreve os principais termos e conceitos utilizados para classificar um sistema como crítico e apresenta propostas, já existentes, de incorporação da Análise de Segurança de software para sistemas críticos. Apresenta as normas mais conhecidas, nas quais o modelo de testes proposto foi embasado, e a evolução dos modelos de desenvolvimento de software.

O terceiro capítulo discorre sobre o processo de testes de sistemas, define os principais tipos de testes, posiciona o trabalho focando na utilização dos testes como forma de prevenir o erro humano no desenvolvimento e na construção de softwares críticos, e demonstra os principais modelos de melhoria de testes.

No quarto capítulo, são apresentados o Modelo de Teste proposto para o planejamento, a construção do projeto de testes e a execução dos testes de

sistemas críticos. No capítulo número cinco, é apresentada a ferramenta desenvolvida para apoiar o modelo proposto, a modelagem das tabelas utilizadas pela ferramenta e as principais telas, com uma breve explicação de cada uma. Um estudo de caso real de sistema crítico é relatado no capítulo seis e as conclusões e possíveis trabalhos futuros constam do sétimo capítulo.

2 SISTEMAS CRÍTICOS E DESENVOLVIMENTO DE SOFTWARE

Neste capítulo é definido sistema crítico, definidos conceitos relacionados a sistemas críticos e teste de software e apresentada uma proposta existente para desenvolver testes para sistemas críticos.

2.1 Introdução

Segundo Ian Sommerville, 2008, Sistemas Críticos são sistemas técnicos ou sociotécnicos dos quais as pessoas ou negócios dependem. Se esses sistemas falharem ao desempenharem os seus serviços conforme o esperado, podem causar sérios problemas e prejuízos significativos. São classificados em três grupos: sistemas críticos de segurança, sistemas críticos de missão e sistemas críticos de negócios.

Desta maneira, no processo de desenvolvimento e manutenção de Sistemas Críticos devem ser utilizadas técnicas, atividades e tarefas que busquem prevenir a ocorrência de acidentes, sendo que quanto maiores ou mais graves forem as perdas ou prejuízos decorrentes de falhas do Sistema Crítico, mais se justificam esforços e recursos investidos na prevenção desses acidentes (modificado de Saeed *et al*., 1991).

Considera-se, neste trabalho, o software como o componente de um computador mais passível à ocorrência de falhas, exigindo a realização de seu

desenvolvimento com Garantia de Segurança, ou seja, garantindo que o mesmo irá funcionar em um sistema sem resultar em riscos inaceitáveis (LOVISI FILHO e CUNHA, 2001).

De acordo com Lovisi Filho e Cunha, 2001, e Rafael Espinha, 2008 a segurança do software envolve estudos preventivos para garantir que o software execute suas funções com resistência às possíveis falhas, preservando a segurança do sistema, dos usuários e do meio ambiente em que está inserido. Pode-se considerar a segurança de software como um fator explícito de qualidade, uma vez que, caso ela não se verifique para um software, ele não atenderá às expectativas do usuário. Segurança é diferente de confiabilidade que também é diferente de disponibilidade

A Garantia de Segurança de Software, segundo (MOURA, 1996) compõe-se de uma sequência de atividades planejadas que visam assegurar a imunidade da aplicação de software em relação à possibilidade de ocasionar acidentes que comprometam a vida humana, o meio ambiente, ou a propriedade. O padrão MIL-STD-498 recomenda a definição de uma Sistemática de Segurança como estratégia de Segurança de Software (SANTELLANO *et. al* ,1998).

Na seção seguinte abordaremos conceitos e na seção 2.3 detalhamos a Análise de segurança de software.

2.2 Termos e Conceitos

Nesta seção apresentamos as definições de alguns termos e conceitos relacionados a testes de software, dependabilidade e segurança que são importantes para sistemas críticos e a proposta feita neste trabalho. Os conceitos apresentados foram derivados dos trabalhos de LAPRIE, 1985, ANDERSON e LEE, 1981, WALLACE e IPPOLITO, 1994, PORTO e DE BARTOLI, 1997 e AVIZIENIS *et al*. ,2004.

Estamos interessados no sucesso de determinado sistema de computação no atendimento da sua especificação. Um defeito (failure) é definido como um desvio da

especificação como percebido pelo usuário. Defeitos não podem ser tolerados, e deve-se fazer o possível para que o sistema não apresente defeitos. Define-se que um sistema está em estado errôneo ou em erro se o processamento posterior a partir desse estado pode levar a defeito. Posto de outra forma, o sistema de informação não está funcionando adequadamente. Finalmente define-se falha (fault ou falta) como a causa física ou algorítmica do erro.

Na figura 2.1 a seguir é mostrada uma simplificação, adotada nesse texto, para os conceitos de falha, erro e defeito. Falhas estão associadas ao universo físico, erros ao universo da informação e defeitos ao universo do usuário. Assim, uma tabela de alocação de lugares em um avião que apresenta uma falha do tipo (falha no universo físico), pode provocar uma interpretação errada da informação armazenada em uma estrutura de dados (erro no universo da informação) e como resultado o sistema pode negar autorização de embarque para todos os passageiros de um vôo (defeito no universo do usuário).

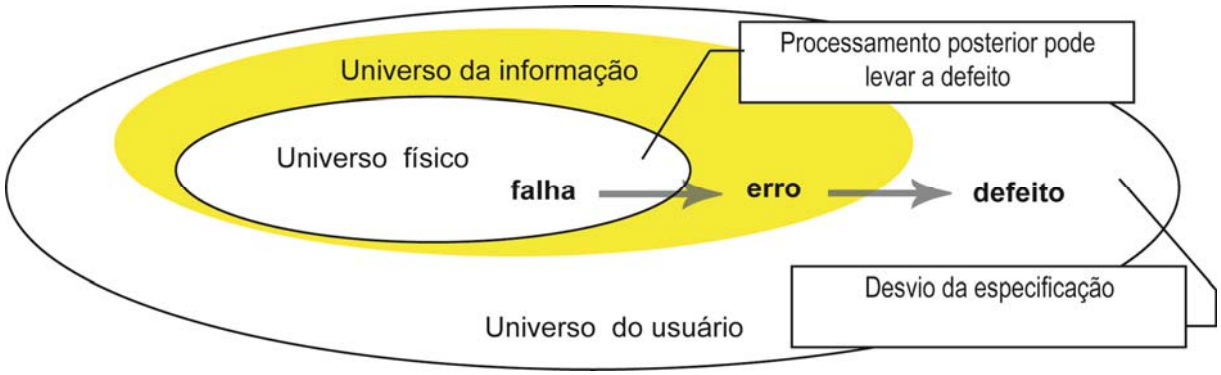


Figura 2.1: Modelo de 3 universos: falha, erro e defeito

Fonte: http://www.inf.ufrgs.br/~taisy/disciplinas/textos/Dependabilidade.pdf

É interessante observar que uma falha não necessariamente leva a um erro (aquela porção da memória pode nunca ser usada) e um erro não necessariamente conduz a um defeito (no exemplo, a informação de vôo lotado poderia eventualmente ser obtida a partir de outros dados redundantes da estrutura). Acidente (Mishap) é um evento não planejado que causa perdas humanas, danos sob sua responsabilidade, ou ainda, a produção de um efeito indesejado do software; Estados Inseguros ou Perigos, Ameaças, (*Threats*) em inglês, são os estados do sistema que, combinados a certas condições externas, podem levar a acidentes;

Para evitar que defeitos apareçam, testes devem ser feitos para encontrar os erros. Assim, testar um software consiste em executá-lo com o propósito de encontrar erros. Quanto melhor planejados forem estes testes, mais abrangentes eles serão e maior a probabilidade de erros serem encontrados. Duas grandes abordagens de teste: caixa branca, onde os testes são planejados considerando-se os código fonte e caixa preta, onde são utilizadas as entradas e as saídas esperadas de acordo com a especificação do software. Para cada abordagem existem inúmeras técnicas. Infelizmente, nenhuma técnica de teste garante que todos os erros foram detectados.

No que tange ao processo de desenvolvimento, duas abordagens estão diretamente ligadas a testes: o processo de verificação e o processo de validação. A Verificação é um processo voltado ao desenvolvedor que procura responder a seguinte pergunta: ‘Estamos fazendo certo o produto?’. Desta forma, procura-se garantir que a forma de trabalhar, as ferramentas, os padrões, entre outros itens, estão sendo utilizados. A Validação é uma abordagem voltada ao usuário/cliente que procura responder a seguinte pergunta: 'Estamos fazendo o produto certo?'. Posto de outra forma, o software atende as expectativas funcionais ou não funcionais do usuário? O foco do presente trabalho situa-se na atividade de Verificação pois a preocupação é garantir que atividades e tarefas projetadas para se garantir a qualidade do software estão sendo executadas

2.3 Análise de Segurança de Software

O ciclo de desenvolvimento de um software compreende todas as fases, que vão desde a concepção até seus ensaios ou testes. Lovisi Filho e Cunha, 2001 propõem a inclusão, nesse ciclo, de um procedimento para a Análise de Segurança de Software. Esse procedimento deve tratar o problema da Segurança de Software desde o início de seu desenvolvimento até sua implementação e produção. O objetivo desse procedimento é determinar e avaliar as falhas do software que possam levar o sistema, do qual o software faz parte, a um estado inseguro.

A Análise de Segurança de Software é de grande importância, principalmente para softwares componentes de Sistemas Críticos. Essa análise é desenvolvida a partir dos requisitos identificados na Análise de Sistemas. Enquanto a fase de Análise

de Sistemas de uma metodologia pretende a identificação das funções que o software deve executar, a Análise de Segurança concentra-se nas que o software não deve executar (LEVESON, 1991).

Para realizar essa análise com sucesso, necessita-se, também, de uma sistemática bem definida, composta de métodos e técnicas, objetivando a identificação e a avaliação dos Estados Inseguros. A seguir apresentam-se as atividades que constituem a Análise de Segurança de Software, de acordo com os estudos publicados nas referências (LEVESON e HARVEY, 1983; LEVESON, 1991; MOURA, 1996).

A Análise de Segurança de Software pretende determinar quais situações podem levar o Sistema a um Estado Inseguro. Suas diversas atividades visam: identificar os Estados Inseguros e as falhas que os originaram; determinar o Fator Crítico (Criticality) dos estados inseguros, citado na seção 2.1; e avaliar a aceitabilidade dos níveis de Segurança do Sistema. Exemplos de Estados Inseguros: o veículo do metrô entrar em movimento com a porta aberta, uma aeronave iniciar um vôo com um equipamento de áudio instalado para avisar à tripulação de um perigo iminente com defeito.

A Figura 2.2 apresenta as atividades que compõe a Análise de Segurança de Software e as informações de entrada e de saída.

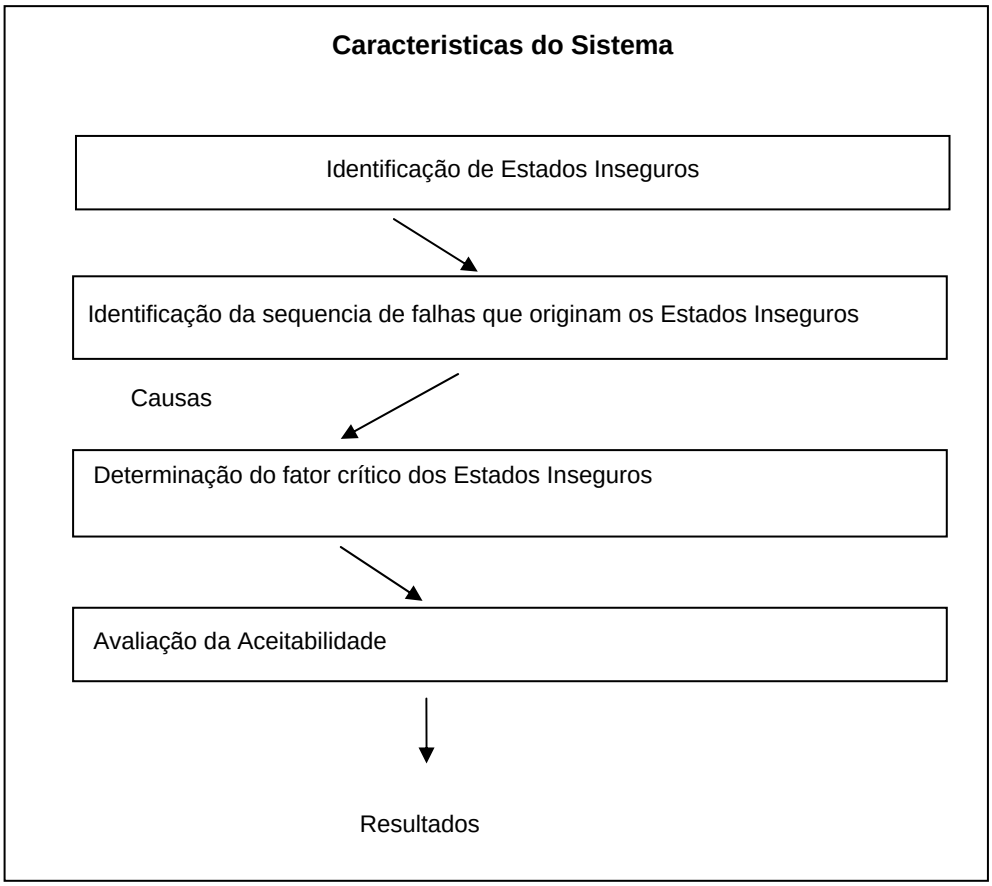


Figura 2.2 - Procedimentos para Análise de Segurança do Software
Fonte: MOURA,1996, p. 78.

No início do desenvolvimento de um Software Crítico, deve ser elaborada a Lista Preliminar de Inseguranças, vide anexo 2, contendo as possíveis falhas para este tipo de software. Essa lista é desenvolvida a partir de informações existentes sobre Análises de Segurança de Software similares. Na medida em que o desenvolvimento avança, realizam-se outras análises, também, para a identificação dos Estados Inseguros e de suas falhas causadoras:

- a Análise Preliminar de Insegurança (Preliminary Hazards Analysis), que visa discriminar os subsistemas críticos e, se possível, propor alternativas de controle, baseia-se nos dados existentes e na experiência dos analistas;
- a Análise de Insegurança de Subsistemas (Subsystem Hazards Analysis) objetiva a identificação dos Estados Inseguros e suas falhas no projeto de cada subsistema e de sua interface. Concentra-se basicamente em aspectos tais como: desempenho, degradação no funcionamento e falhas funcionais, procurando determinar os modos de falhas e seus efeitos;

- a Análise de Insegurança de Sistema (*System Hazards Analysis*) identifica os Estados Inseguros e suas falhas, associados às interfaces entre os subsistemas, incluindo erros potenciais humanos; e

a Análise de Insegurança na Operação e Suporte (*Operation and Support Hazard Analysis*) avalia os Estados Inseguros ocorridos durante as etapas de uso e manutenção do sistema, especialmente aqueles provenientes da interação do homem com o Sistema. Para realizar essas análises, é necessária a utilização de algumas técnicas para identificação de Estados Inseguros, tais como: Hazop, SFTA, FMECA e Redes Petri Temporais (IPPOLITO e WALLACE, 2009).

Após a identificação dos Estados Inseguros, realiza-se uma avaliação quanto ao seu Fator Crítico. Fator Crítico (FC) é uma classificação proposta por Minister of Defense, 2000, em que cada perigo ou risco identificado em um sistema crítico recebe uma classificação variável de T1 a T4 em função de probabilidade de sua ocorrência e da gravidade de suas conseqüências (ver Quadros 2.1 e 2.2).

Vários fatores, como tempo, possibilidade de perda de grandes volumes financeiros, capacidade dos equipamentos, confidencialidade dos dados e outros, impactam sistemas críticos. Um Sistema Crítico (Critical System) apresenta restrições mais fortes relacionadas a um determinado fator. Alguns sistemas podem apresentar restrições a mais de um fator. O levantamento inicial dos fatores de criticidade vai determinar qual é o fator ou quais são os fatores mais vulneráveis do sistema. A determinação do Fator Crítico dos Estados Inseguros consiste em classificar os mesmos, de acordo com a Severidade (Severity) do acidente causado por eles (Catastrófico, Crítico, Marginal e Desprezível) e com a Probabilidade (Probability) de ocorrência deste acidente (Frequente, Provável, Ocasional, Remota e Improvável). O Quadro 2.1 mostra a classificação dos Estados Inseguros quanto à severidade e o Quadro 2.2 determina o Fator de Criticidade conforme o grau de severidade e a probabilidade de ocorrência.

Categoria	Classificação da Categoria	Conseqüências Possíveis
Catastrófico	I	Pode resultar em morte, invalidez total permanente, perdas excedentes a US$ 1.000.000,00, ou danos que violem a lei ou as regulamentações.
Crítico	II	Pode resultar em invalidez parcial ou permanente, lesões ou doenças ocupacionais, que requeiram hospitalizações de, pelo menos três pessoas, danos materiais entre US$ 200.000,00 e US$ 1.000.000,00 ou danos ambientais reversíveis que causem violação de leis e regulamentações.
Moderada	III	Pode resultar em lesões ou doenças ocupacionais, resultando em um ou mais dias de trabalho perdidos, perdas materiais entre US$ 100.000,00 e US$ 200.000,00 ou danos ambientais sem violação de lei ou regulamentação, em que as atividades de restauração poderiam ser executadas.
Insignificante	IV	Pode resultar em lesões ou doenças que não causem perdas de dias trabalhados; perdas entre US$ 2.000,00 e US$ 10.000,00 ou danos ambientais que não violem a lei ou a regulamentações ambientais.

Quadro 2.1 - Classificação do Acidente quanto a severidade
Fonte: MINISTRY OF DEFENSE, 2000.

Fator Crítico				
	Categoria			
Faixa de probabilidade	Catastrófica I	Crítica II	Moderada III	Insignificante IV
Frequente	T4	T4	T3	T2
Provável	T4	T3	T3	T2
Ocasional	T3	T3	T2	T2
Remota	T3	T2	T2	T1
Improvável	T2	T2	T1	T1

Quadro 2.2 - Classificação do Fator Severidade X Probabilidade
Fonte: MINISTRY OF DEFENSE, 2000.

Cada nível de Fator Crítico levantado deve receber um tratamento diferenciado para garantir a Segurança do Software. Um Fator Crítico em um estado do nível T4 necessita de atenção especial, exigindo a utilização de estruturas de programação para evitar, controlar ou recuperar sua ocorrência, ao passo que, um estado do nível T1 pode não demandar tantos cuidados. Deve-se observar, também, que, devido à falta de dados numéricos para o software, geralmente desconsidera-se sua classificação quanto à probabilidade. Após o término da determinação do Fator

Crítico, avaliam-se os Estados Inseguros não evitados, controlados ou recuperados completamente, para determinar se os mesmos realmente não representam uma ameaça à Segurança do Software, garantindo, assim, um nível aceitável de risco.

2.4 Normas utilizadas em Sistemas Críticos

As normas desempenham importantes papéis na execução de um projeto, principalmente em Sistemas Críticos. Elas servem de auxílio à equipe de projeto, garantindo que o produto em desenvolvimento obedeça a um determinado nível mínimo de qualidade, na seleção de métodos de projeto de eficiência reconhecida. Também promovem uniformidade entre as diversas equipes de trabalho, proveem guias de projeto, além de proporcionar uma base legal no caso de disputas judiciais.

Para certificar um sistema, se faz necessária a utilização de normas apropriadas a cada aplicação. Algumas são genéricas e outras se aplicam a casos particulares. Este trabalho apoia-se no Projeto SPICE, Norma ISO/IEC 15504 e na NBR ISO/IEC 12207, 1995 (Processos de Ciclo de Vida de Software). O modelo proposto visa atender a norma 12207 no item 'Processos de Apoio', no processo de “Garantia da Qualidade” (QUA), itens QUA.2 - Verificação e QUA.3 – Validação, já que Verificação e Validação podem ser feitas utilizando testes. Atende também ao módulo 'Processos Fundamentais' no Grupo Engenharia (ENG), ENG 10. Esse item do grupo cita, explicitamente, a atividade ‘Teste de Sistema’, conforme a Figura 2.2:

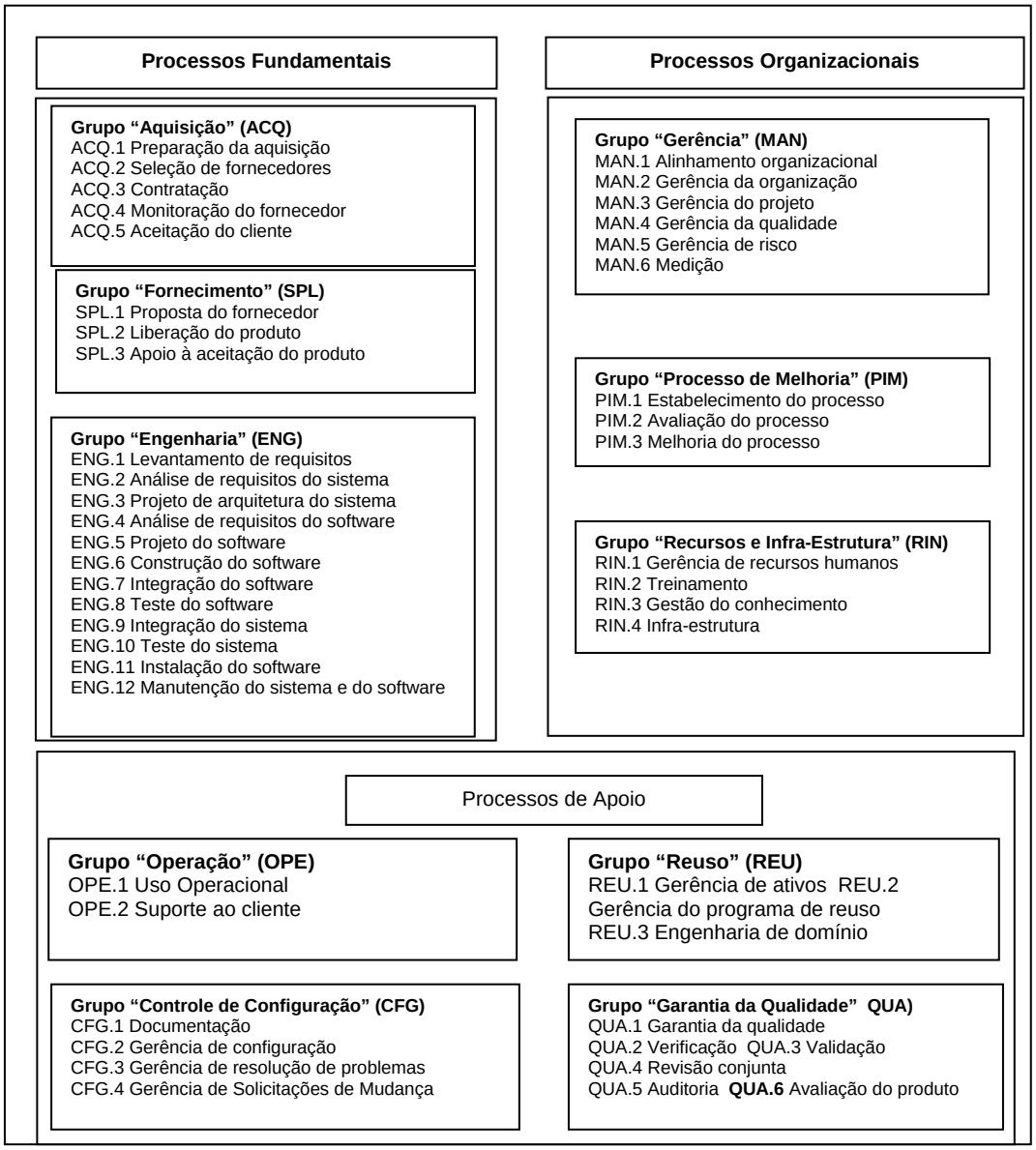


Figura 2.3 - Arquitetura da ISO 12207 com FDAM-1 e FDAM-2
Fonte: ESTOLANO, 2005.

A norma IEEE Std 829, 1998 determina a produção de três documentos na atividade especificação de testes: Especificação de Projeto de Teste, Especificação de Caso de Teste e Especificação de Procedimento de Teste. Os relatórios de teste são cobertos por quatro documentos: Diário de Teste, Relatório de Incidente de Teste, Relatório-Resumo de Teste e Relatório de Encaminhamento de Item de Teste.

A norma separa as atividades de teste em três etapas: preparação do teste, execução do teste e registro do teste. Embora a norma possa ser utilizada para o teste de produtos de software de qualquer tamanho ou complexidade, projetos pequenos ou de baixa complexidade podem agrupar alguns documentos propostos, diminuindo o gerenciamento e os custos de produção dos documentos. O conteúdo

dos documentos também pode ser abreviado. Adicionalmente, a equipe responsável pelo teste deverá tomar outras decisões em relação à aplicação da norma em projetos específicos, decidindo, por exemplo, se é mais conveniente elaborar um único plano que englobe os testes de unidade, integração e aceitação, ou um plano para cada uma das fases de teste citadas.

Mais do que apresentar um conjunto de documentos, que deve ser utilizado ou adaptado para determinadas empresas ou projetos, a norma apresenta um conjunto de informações necessárias para o teste de produtos de software. Sua correta utilização auxiliará a gerência a se concentrar, tanto com as fases de planejamento e projeto, quanto com a fase de realização de testes propriamente dita, evitando a perigosa armadilha de só se pensar nos teste de um produto de software após a conclusão da fase de codificação.

2. 5 Modelos de Desenvolvimento e Testes

2.5.1 Os Modelos Cascata, Espiral e Modelo 'V' e como cada aborda os testes

O modelo Cascata foi um dos primeiros modelos de ciclo de vida (BOEHM, 1981). Nesse modelo cada uma das etapas deve ser finalizada antes da etapa seguinte ser iniciada. O processo flui sempre para frente. Fica claro que a atividade de testes se inicia após a construção do software. A preparação para os testes não é clara nesse modelo e os procedimentos para remoção das falhas exigem o retorno à fase de desenvolvimento. O teste, como a última atividade antes da implantação, corre o risco de ser resumido e, às vezes, até omitido completamente.

O Modelo Espiral proposto por Boehm em 1988, apresenta uma visão conjuntural do processo de desenvolvimento e também a visão da prototipação. Aparece a atividade de Análise de Riscos, o conceito de validação dos requisitos, a fase de testes, a integração e a atividade de aceitação. Entretanto, esse modelo também prega que os testes devem ser feitos após a construção do sistema e, também, não identifica as atividades necessárias à remoção de defeitos. O modelo ‘V’, evolui um pouco, incorporando essas atividades. Conceitualmente, o Modelo 'V' foi desenvolvido simultaneamente, porém independentemente, na Alemanha e nos Estados Unidos, no final dos anos 80. Nos Estados Unidos, foi documentado em 1991, antes do National Council on Systems Engineering (NCOSE), (NCOSE, 1991).

O modelo foi utilizado para desenvolver sistemas de controle de satélites nas camadas de hardware, software e interação humana. Atualmente, atende a aplicações comerciais tão bem quanto a aplicações críticas. É muito utilizado em gerenciamento, durante o ciclo de vida de projetos. Na figura 2.4, é mostrado o modelo 'V' focando os processos de desenvolvimento e de testes conjunto. Os testes começam a ser reconhecidos e alçados a um patamar de maior importância (FWHA, 2005).

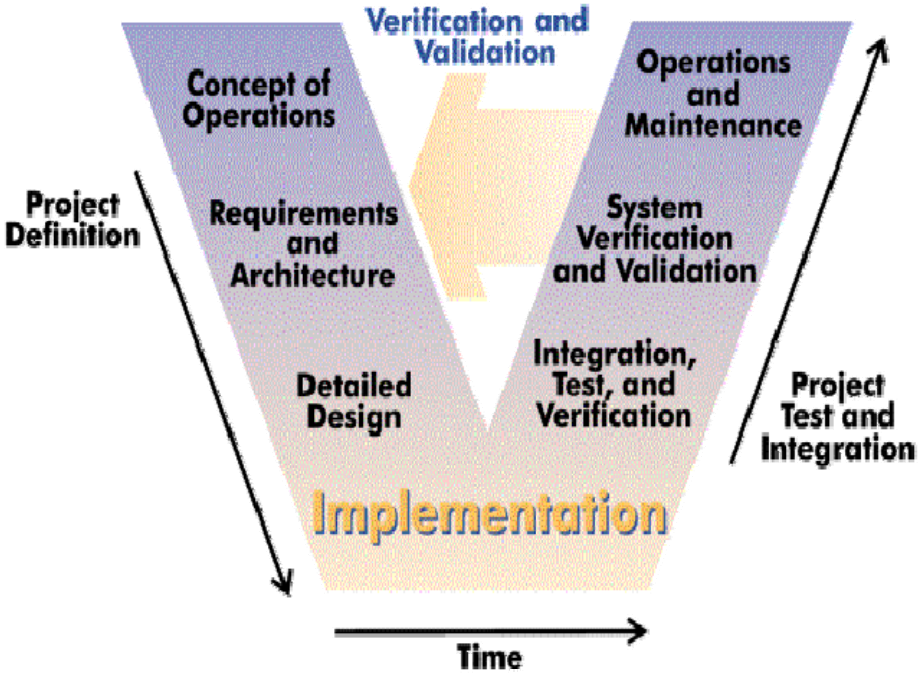


Figura 2.4 - Modelo 'V' convivência desenvolvimento e testes
Fonte: FHWA, 2005.

No modelo 'V' também é possível alocar os atores, analistas de desenvolvimento e de teste, exercendo, cada um, suas atividades, porém juntos em cada etapa do ciclo de vida do software. A Figura 2.5 situa esses atores no modelo, focando a importância dos testes, principalmente para as aplicações críticas.

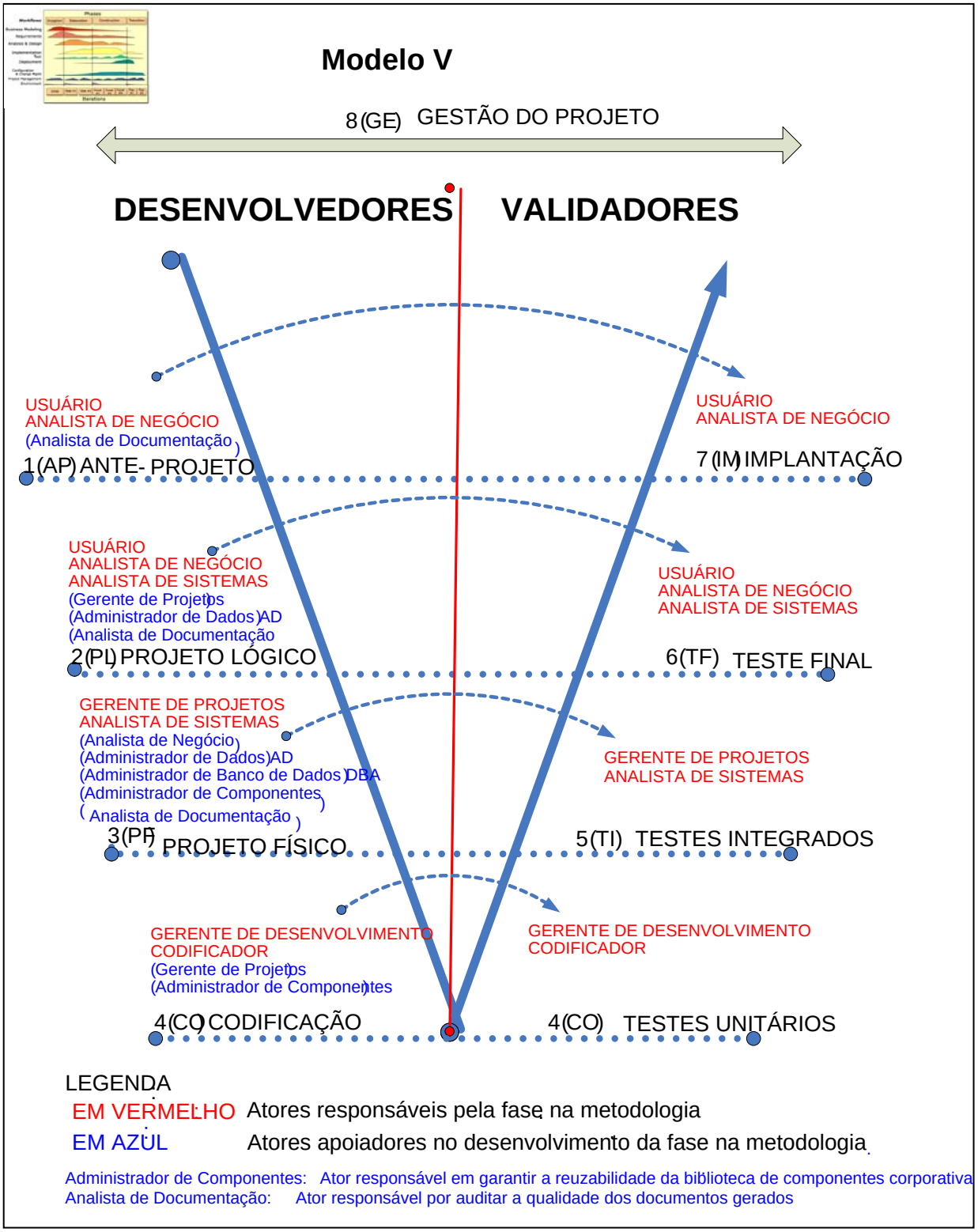


Figura 2.5 - Modelo 'V' Atividades de Desenvolvimento e Testes com os principais atores escalados
Fonte: UALG, 2009.

Ualg (2009) argumenta que o modelo ‘V’ permite um entendimento comum e simultâneo das fases por parte dos desenvolvedores e clientes, uma vez que os testes não são executados somente no fim do ciclo de desenvolvimento (Figura 2.6).

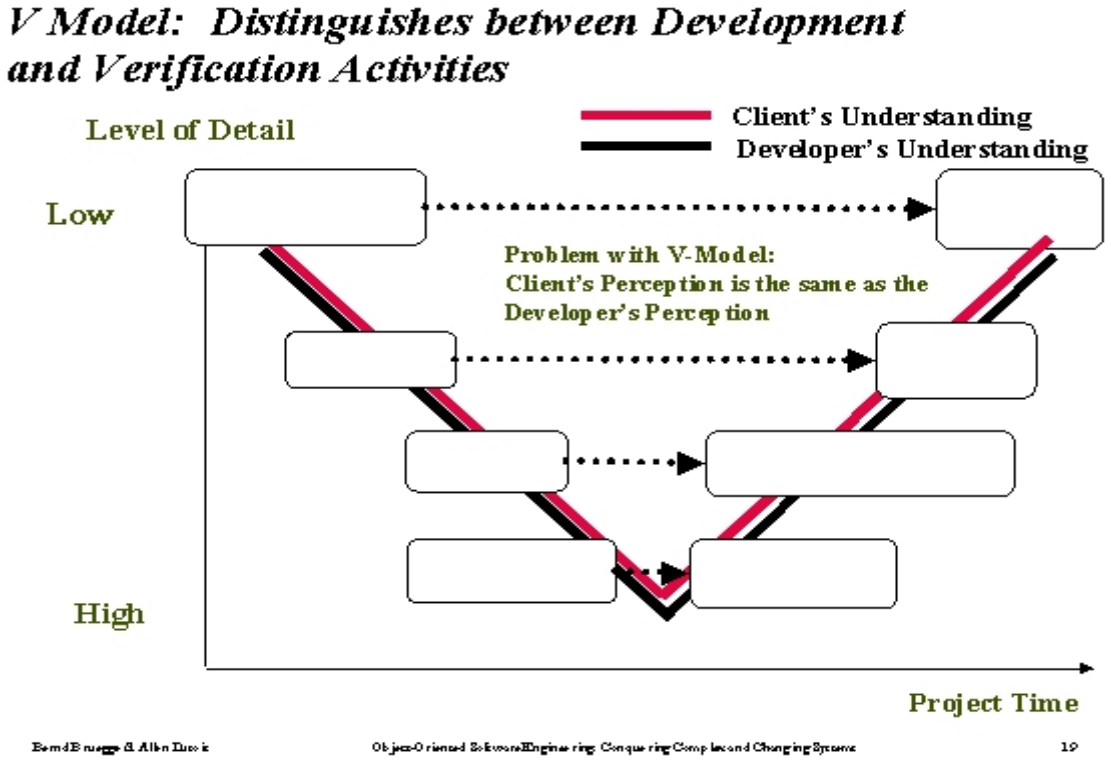


Figura 2.6 Modelo 'V' Atividades de Desenvolvimento X Atividades de Verificação
Fonte: UALG, 2009.

São vantagens do modelo ‘V’: a fase de teste começa no início do ciclo, os Planos de Testes detalhados em cada fase do ciclo ajudam a compreender melhor qual a origem do problema. O modelo ‘V’ é muito utilizado para o desenvolvimento de sistemas IT, sendo superior ao modelo cascata e ao modelo espiral na construção de projetos de IT (UALG, 2009). Mas também foram identificadas em trabalhos da Universidade de Algarves em Portugal algumas desvantagens do modelo 'V':

- a percepção do cliente e do desenvolvedor é a de que o modelo continua a não ser suficientemente flexível.
- é necessário maior *feedback* entre todas as fases do ciclo.

Uma característica importante desse modelo é que tempo e maturidade movem-se da esquerda para a direita e existe interação entre os níveis mais baixos e mais altos da hierarquia do sistema, como pode ser visto na Figura 2.6. Esse modelo coloca a verificação e validação como uma atividade a ser desenvolvida durante todo

o processo de desenvolvimento. O próprio nome do modelo remete às atividades de 'Verificação' e 'Validação' (UALG,2009).

Um primeiro olhar dá a impressão da mesma sequência de atividades dos outros modelos de construir primeiro para depois testar, mas a colocação das atividades de V&V em cada etapa demanda um trabalho preparatório que diminui o distanciamento entre a construção e o teste. Por exemplo, elaborar o plano de teste e testar a estratégia de testes é um procedimento que deve ser realizado logo após a identificação dos requisitos. Porém, falta nesse modelo um planejamento claro para a remoção de defeitos e execução de testes de integração e regressão. No caso de sistemas críticos, parece-nos que falta o controle mais apurado sobre os requisitos críticos do sistema a ser desenvolvido.

Do ponto de vista dos testes, todos os modelos apresentados têm deficiências: o teste é sempre iniciado após a codificação, não fica claro como fazer os testes entre as várias fases do sistema, testes mais rígidos, dependendo da criticidade e dos perigos aos quais o sistema está exposto não são mencionados. O modelo ‘W’, que se segue, entretanto, já exibe uma tendência de fazer fronte a alguns desses problemas, removendo algumas dessas desvantagens.

2.5.2 O modelo W

No desenvolvimento de softwares de 30 a 40 por cento das atividades estão relacionadas com testes. Por isso, é crucial que se iniciem as atividades de teste no início do projeto e não deixá-las para depois da codificação ser concluída. Apesar de novos modelos de desenvolvimento de software como o Rational Unified Process (RUP) e o Extreme Programing (XP) serem muito populares entre os engenheiros de software e do Modelo 'V' ter ganhado muita aceitação, o modelo 'W', proposto por Spillner (2002), surgiu com a intenção de colocar os testes em condições de igualdade com as demais tarefas do desenvolvimento. Ele se baseia no modelo 'V' para mostrar a importância e a dinâmica da atividade de testes. Um segundo 'V' é adicionado ao modelo. Os dois V's dão origem ao modelo 'W'.

O modelo 'W' associa as tarefas de testes diretamente a cada fase de desenvolvimento descrita no modelo ‘V’ e esclarece a prioridade das tarefas e a

dependência entre as atividades de teste e as de desenvolvimento. Além disso, esclarece que teste e depuração não são a mesma coisa (SPILLNER, 2002).

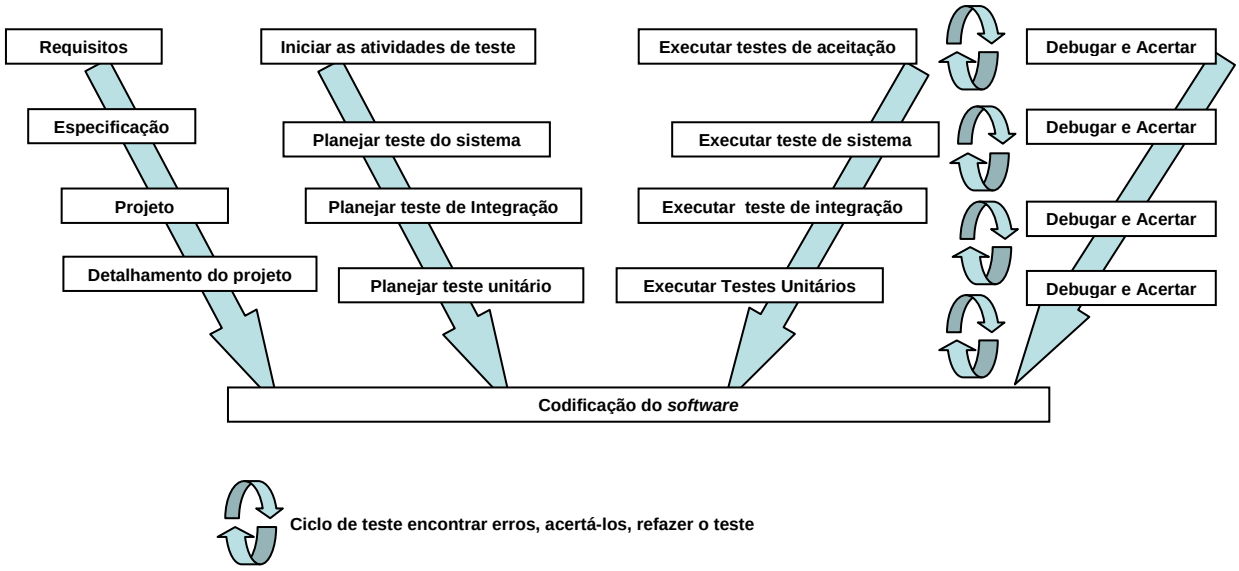


Figura 2.7 - Modelo `W`
Fonte: SPILLNER, 2002.

De acordo com o modelo 'W', os testes do sistema devem iniciar-se já nas primeiras etapas do desenvolvimento. Nessa etapa não é só o planejamento e gestão das tarefas que tem de ser realizadas. O sistema é dividido em componentes e processos para serem testados. Os componentes devem ser desenvolvidos com o foco na viabilização dos testes.

Quando os requisitos de um sistema estão claros, os casos de uso construídos e bem entendidos, os dados e o conhecimento para a confecção dos casos de teste estão à mão, então, o esforço para criar os testes é menor do que se estas atividades forem adiadas para outra etapa.

Para cada etapa, os testes executados devem levar a descoberta de defeitos. Os defeitos devem ser localizados e mudanças devem ser implementadas para eliminar os defeitos e, finalmente, o teste deve ser reexecutado, até que o resultado seja correto.

2.6 Planejamento de Testes de Sistemas Críticos conforme o Modelo ‘W’

Em se tratando de sistemas críticos, ao iniciar-se a atividade de planejamento dos testes, devem ser feitos, a revisão dos requisitos, a homologação oficial dos mesmos com os especialistas do domínio e usuários responsáveis/homologadores, e o planejamento e a elaboração de testes de aceitação. As atividades de testes que devem começar no início do projeto são:

- fixação da estratégia de teste e conceitos de testes;
- análise de risco, levantamento das ameaças , identificação das criticidades e os FC's, de cada ameaça inventariada, estimar os gastos com os testes, recursos humanos, recursos de máquina e tempo necessário para executar os testes;
- elaborar o plano de testes;
- treinar a equipe de testes, se necessário;
- estabelecer processos e métricas de controle;
- definir recursos de hardware, computadores, mesas, cadeiras, telefones, etc.; e
- providenciar recursos de software, Bases de Dados (definir bases de teste), confidencialidade, confiabilidade, coerência, consistência e políticas de atualização e recuperação periódica, controle de versão, ferramentas de testes, etc.

Essas atividades são as bases para permitir uma gerência de alta qualidade nas atividades de teste. A estratégia de testes é determinada após uma cuidadosa análise de risco que vai determinar os recursos a serem gastos no planejamento do plano de testes do sistema. Durante o desenvolvimento dos sistemas, todos os planos devem ser controlados e atualizados, todas as decisões documentadas e a necessidade de rastrear as mudanças com os nomes dos responsáveis (e-mails, atas de reuniões, solicitações de alterações). Quando se trabalha debaixo de um processo de desenvolvimento maduro, as inspeções são realizadas durante todo o processo. Isso é essencial, quando se trata de sistemas críticos.

As revisões devem responder a perguntas como: 'todos os requisitos levantados estão sendo testados?', 'o risco de cada um deles foi exaustivamente

levantado?', 'o levantamento está completo e consistente?'. Deve-se sempre olhar para trás para corrigir problemas, antes de seguir em frente para o desenvolvimento. Mas tão importante quanto a atenção com o passado é o olhar para frente. Devem ser feitas perguntas tais como: Os requisitos críticos são testáveis? Os custos previstos são defensáveis? Se as respostas forem claras, então não haverá problemas para que esses requisitos sejam implementados. Se não existir um caminho claro de como os requisitos vão ser testados, então é provável que não se tenha idéia de como fazer para implementar esses requisitos (SPILLNER, 2002).

Na fase inicial de levantamento e identificação de requisitos, o conhecimento para a construção dos testes de aceitação está disponível e à mão. Esse é o melhor momento para fazer o planejamento e a construção dos casos de teste. Nesse momento, é possível estabelecer as prioridades dos testes em função dos fatores de criticidade dos sistemas críticos, especificar requisitos funcionais e não funcionais e, se possível, providenciar toda a infraestrutura para execução dos testes. No próximo capítulo, passaremos a detalhar os testes dos sistemas.

3 O PROCESSO DE TESTES DE SISTEMAS E MODELOS DE MELHORIA

Este capítulo apresenta uma visão geral dos conceitos e processos testes de software dentro dos modelos de qualidade. A introdução mostra os objetivos do teste e suas funções dentro da atividade de desenvolvimento de software. Cita os vários tipos de testes adequados a cada fase. Salienta a importância dos testes para sistemas críticos e reforça a abordagem que eles devem estar presentes em todas as fases do ciclo de desenvolvimento. São relacionados alguns tipos de testes que consideramos mais importantes e mais utilizados. Foca-se a inevitabilidade do erro humano e a tarefa de testar. Relaciona-se alguns dos principais modelos de melhoria da atividade de teste e finaliza com orientações sobre como prevenir os erros e otimizar o controle de qualidade, utilizando o teste de software.

3.1 Introdução

O teste do software é uma das fases do processo de Engenharia de Software que visa atingir um nível superior de qualidade de produto. Testar um software significa verificar, através de uma execução controlada, se o seu comportamento se dá de acordo com o especificado. O objetivo principal dessa tarefa é encontrar o número máximo de erros, dispondo do mínimo de esforço, mostrando se os resultados estão, ou não, de acordo com os padrões estabelecidos (DIAS, 2009). Embora muitos considerem que o teste de software serve para demonstrar o correto funcionamento de um programa, na verdade, ele é utilizado como um processo da Engenharia de Software para identificar defeitos. Testes podem ser abordados do ponto de vista de técnicos (seção 3.2) ou processos (seção 3.5).

3.2 Os Testes no Desenvolvimento dos Sistemas

Durante o processo de desenvolvimento, o software se transforma com o avanço das atividades, requisitos, protótipos, modelo de dados lógico, modelo de dados físico, código-fonte, módulos funcionais. As atividades a serem desenvolvidas e as fases a serem cumpridas e gerenciadas durante o desenvolvimento de um sistema fazem com que, na maioria das vezes, o teste seja relegado a segundo plano no desenvolvimento dos sistemas convencionais. É comum que as equipes só se dediquem as atividades de testes depois que os programas já estão codificados. Muitas vezes a pressão para cumprir os prazos dos cronogramas de implantação faz com que a fase de testes seja abreviada e restrita ao tempo que se tem disponível.

O ideal é que os testes sejam executados desde o início do desenvolvimento do sistema. No caso de sistemas críticos, isso é essencial. Mesmo que o sistema já esteja completamente desenvolvido, os COTS, *Commercial Off-The-Shelf systems,*. chamados 'produtos de prateleira', e o projeto seja de sua implantação ou customização, os testes devem ser executados para comprovar o atendimento aos requisitos levantados. O Processo de Testes deve ser adaptado a essa situação e o ideal é que os testes sejam feitos durante todo o processo de desenvolvimento (PAULA, 2007). A Figura 3.1 mostra, segundo Pfleeger (2004), os tipos de teste que devem deve ser executados em cada fase de desenvolvimento de um software. Essa estratégia contribui para aumentar a confiança na exatidão do projeto, quando o sistema é um sistema crítico.

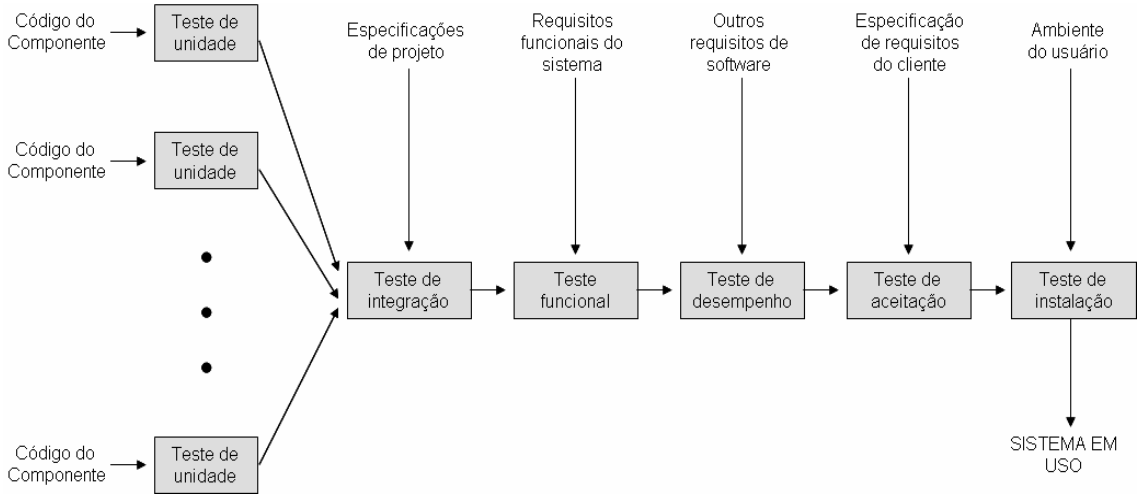


Figura 3.1 - Tipos de Testes e Fases de Desenvolvimento Fonte: PFLEEGER, 2004.

Independente do tipo de teste e do momento que eles são executados os testes seguem os passos:

- formular um caso de teste segundo um critério estabelecido;
- determinar os resultados esperados dos casos de teste;
- rever os casos de teste e seus resultados esperados;
- comparar os resultados do teste com o resultado esperado; e
- explicar os problemas encontrados de forma que possam ser corrigidos no sistema.

3.3 Tipos de Testes

Existem vários tipos de testes e cada um deles é mais adequado a cada situação. A equipe de testes deve escolher o mais adequado ao sistema e à fase a ser testada, sempre considerando os requisitos do sistema. Detalhamos no quadro 3.1 alguns tipos de testes descritos por Myers (1979), Hetzel (1988) e Beizer (1995). Existem muitos outros, mas como este trabalho foca o processo e não as técnicas de teste, não pretendemos esgotar este detalhamento.

Testes Caixa Preta (Black Box)	Verificam a funcionalidade e a aderência aos requisitos, em uma visão externa ou do usuário, sem conhecimento do código e da lógica interna do componente testado.
Testes Caixa Branca (White Box)	Avaliam as cláusulas de código, a lógica interna do componente codificado, as configurações e outros elementos técnicos.
Testes Unitários	São aplicados nos menores componentes de código.
Testes de Integração	Verificam se o software continua se comunicando com outros softwares.
Testes de Regressão	Visam garantir que o software permaneça intacto depois de alterações efetuadas.
Testes de Carga	Avaliam a resposta de um software sob uma pesada carga de dados, repetição de certas ações de entrada de dados, entrada de valores numéricos grandes, consultas complexas a base de dados,
Testes Back-to-back	Executar o mesmo teste em versões diferentes do software e os resultados são comparados.
Testes de Configuração	Verificam se o software está apto a funcionar em diferentes versões ou configurações de ambientes (hardware e software).
Testes de Usabilidade	Verificam o nível de facilidade de uso do software pelos usuários.
Testes de Instalação	Verificam o processo de instalação parcial, total ou atualização do software.
Testes de Segurança	Validam a capacidade de proteção do software contra acessos interno ou externo não autorizado.

Quadro 3.1 - Tipos de Testes Fonte: MYERS, 1979; HETZEL, 1988 e BEIZER, 1995.

Em se tratando de Sistemas Críticos, os testes devem ser utilizados exaustivamente. Os artefatos que irão minimizar os perigos e lidar com as situações de riscos identificados para o sistema em desenvolvimento ou manutenção, precisam ter rastreabilidade. No modelo proposto eles são chamados de artefatos de missão crítica e passarão por testes mais robustos e exaustivos com o objetivo de encontrar o maior número de inconsistências possíveis.

3.4 Os Testes e o Erro Humano

O erro humano pode ser fatal se o sistema é critico. Como não é possível impedir o erro humano, é necessário conviver com ele e neutralizá-lo, sempre que possível. O teste e uma das maneiras de se fazer isso.

Apesar dos avanços da Engenharia de Software, o desenvolvimento de software, ainda é extremamente dependente da atividade humana. As pessoas ainda estão envolvidas em todas as etapas de desenvolvimento de software, desde a

identificação de requisitos e construção do sistema, até a homologação do produto. Os modelos propostos e utilizados pela indústria de software são centrados em processos. Os mecanismos por eles estabelecidos para a melhoria da qualidade de software geralmente só consideram o ser humano quando tratam do treinamento das equipes (SANDHOF e FILGUEIRAS, 2007).

A confiabilidade da operação humana é motivo de estudos e preocupação durante toda a vida do sistema. Existe uma irrefutável certeza de que erros humanos acontecem por diversas causas. Se os erros não forem detectados podem causar grandes perdas (SANDHOF e FILGUEIRAS, 2007).

Este trabalho, que trata os Sistemas Críticos, defende que o possível erro deve ser identificado antes de sua ocorrência e processos de blindagem devem ser implantados para neutralizar o seu efeito. Nesse tipo de sistema deve-se ter em mente que as perdas possíveis de serem evitadas não podem ocorrer.

Brian Marick em 1997, resumiu em um artigo os principais erros que são cometidos, mesmo quando se opta por executar formal e planejadamente os testes em softwares durante seu desenvolvimento e manutenção. Foram identificados cinco grupos importantes de erros comumente cometidos por quem testa software, nos seguintes pontos:

1 - No propósito da atividade de teste: ocorre quando o ator que controla a execução não entende bem qual o sentido de se fazer a atividade de testar e não aproveita os resultados de forma eficaz. Os erros comuns são:

- Atribuir a responsabilidade pela qualidade unicamente à equipe de teste;
- Achar que a tarefa de equipe de testes é simplesmente encontrar erros;
- Não encontrar os erros importantes;
- Não informar sobre erros de usabilidade;
- Não focar em estimar a qualidade; e
- Oferecer estatísticas de erros sem o contexto relevante.

2 - No planejamento dos testes: esses erros são relacionados à fase de planejamento dos testes:

- Concentrar exageradamente em teste funcional;
- Não enfatizar o teste de configuração;
- Deixar o teste de carga para o final do processo;
- Não testar a documentação e não testar a instalação;

- Confiar exageradamente no teste beta;
- Executar os testes de forma estritamente consecutiva;
- Não identificar áreas de risco; e
- Teimosia em aplicar um plano de teste ineficaz.

3 - No pessoal contratado para testar. Ao montar a equipe que conduzirá os testes, é importante se concentrar nos seguintes erros comuns e tentar evitá-los:

- Usar o teste como um trabalho temporário para programadores novos;
- Recrutar ex-programadores (os que não são os melhores) para a equipe de teste;
- Usar testadores que não conhecem o domínio da aplicação;
- Não contratar pessoal de suporte técnico e documentação;
- Insistir que testadores sejam programadores;
- Não utilizar uma equipe de teste diversificada;
- Separar fisicamente testadores e programadores;
- Acreditar que programadores não podem testar seu próprio código;
- Não treinar e nem motivar os programadores para testar.

4 - Na execução dos testes em si. Durante a execução dos testes, um conjunto de problemas comuns é:

- Dar mais atenção à execução dos testes do que ao seu projeto;
- Não rever os projetos de teste;
- Ser específico demais nas entradas e procedimentos do teste;
- Não notar e nem explorar irregularidades peculiares;
- Checar que o produto faz o que deve, mas não verificar se ele não faz o que não deve;
- Elaborar suítes de teste compreendidas apenas por seus criadores;
- Testar apenas através da interface com o usuário;
- Informar problemas ineficiente ou incompletamente;
- Adicionar apenas testes de regressão quando problemas são encontrados; e
- Não manter um histórico de anotações para os próximos testes.

5 - Na aplicação da tecnologia nos testes: a aplicação de tecnologia à atividade de teste é muitas vezes benéfica, mas nem sempre e nunca desenfreadamente. Abaixo os erros que envolvem o foco tecnológico do teste:

- Tentar automatizar todos os testes;
- Esperar reexecutar testes manuais;
- Usar ferramentas de automação de interface para reduzir o custo do teste;
- Esperar que os testes de regressão encontrem uma grande proporção dos defeitos introduzidos;
- Abraçar exageradamente a análise de cobertura do código;
- Remover testes da suíte de testes de regressão porque não aumentam a cobertura;
- Usar cobertura como um objetivo primário para a equipe de teste; e
- Não analisar a cobertura de código em absoluto.

Neste trabalho, ao propor o processo para testes de sistemas críticos, seremos cuidadosos para tentar evitar todos os erros clássicos descritos por Marick.

3.5 Modelos de Melhoria de Software e de Testes

Para o desenvolvimento de software com qualidade, dentro de prazos e custos controlados e compatíveis com o mercado, é fundamental a melhoria dos processos da engenharia de software. Para alcançar esse patamar, modelos de melhoria da qualidade tem sido utilizados com sucesso pelas organizações de software. Os modelos mais utilizados são: ISO/IEC 12207, ISO/IEC 15504, CMMI e MPS.BR.

Esses modelos identificam processos fundamentais para a Engenharia de Software. Todos eles citam, direta ou indiretamente, o teste de software como um desses processos. O Teste é fundamental para a avaliação do software desenvolvido. Testar software não é uma atividade trivial e exige conhecimentos, habilidades e infraestrutura específicos. Trata-se de uma atividade tão importante que, recentemente, surgiu uma proposta específica para o processo de teste alinhada com o MPS.BR, é o MPT.BR (MPT.BR , 2009).

A metodologia do teste de software está impactando o comportamento das empresas na busca em implantar e melhorar o processo de teste utilizado. Ainda que as técnicas de teste de software mais utilizadas tenham sido criadas por volta dos

anos 70, as empresas tem uma grande dificuldade com a atividade de teste. Isso pode ser um reflexo da falta de profissionais especializados na área de teste de software, ou mesmo da dificuldade em implantar um processo de teste, utilizando as técnicas existentes na literatura.

Os modelos de melhoria de software citam a atividade testes sem, contudo detalhá-la. Os modelos de melhoria dos testes focam no processo de testes e fornecem orientações para que essa disciplina seja mais planejada e os recursos nela investidos tenham o maior retorno possível. São apresentados resumidamente os modelos MPS.BR, CMMI, *Capability Maturity Model Integration*, o TMM, *Test Maturity Model* e o MPT.BR, Melhoria de Processo de Teste Brasileiro.

O MPS.BR, Modelo Brasileiro Para Melhoria do Processo de Software, baseia-se nos conceitos de maturidade e capacidade de processo para a avaliação e melhoria da qualidade e produtividade de produtos de software e serviços correlatos (MPS.BR, 2009). É constituído de três componentes: o Modelo de Referência (MR), o Modelo de Avaliação (MA) e o Modelo de Negócios (MN). Esses modelos foram construídos segundo as premissas da ISSO/IEC 12207 ISSO/IEC 15504 e CMMI. O Modelo define sete níveis de maturidade, constituídos de duas dimensões cada um: Processos e Atributos do Processo (AP), vide Tabela 3.1, Níveis do MPS.BR Processos e Capacidades (MONTONI *et al.*, 2007).

O cenário do Modelo de Testes de Sistemas Críticos que apresentamos situa-se no mínimo no nível ‘D’ do MPS.BR, em que encontram-se as atividade de Verificação (VER) e Validação (VAL) e as capacidades AP 1.1 Gerência de Projetos - GPR e AP 2.1 Gerência de Risco – GRI. Ao propor uma metodologia de testes aderente ao Modelo MPS.BR, vislumbramos utilizar o impulso criado com o esforço de capacitação despendido pela Softex, Organizações e Profissionais. O impacto tecnológico e mudança cultural provocados pela disseminação do modelo em todas as regiões do Brasil, permitirá mobilizar a Engenharia de Software em direção à segurança que o desenvolvimento da disciplina Engenharia de Testes pode proporcionar, principalmente em se tratando de Sistemas Críticos.

Nível	Processo	Capacidade
A – Em otimização (Mais Alto)	Analise de causas e problemas de resolução - ACP	AP 1.1, AP 2.1, AP 2.2, AP 3.1, AP 3.2, AP 4.1, AP 4.2, AP 5.1 e AP 5.2
B – Gerenciado Quantativamente	Gerencia de Projetos – GPR (Evolução)	AP 1.1, AP 2.1, AP 2.2, AP 3.1, AP 3.2, AP 4.1, AP 4.2
C - Definido	Gerencia de Riscos – GRI Desenvolvimento para Reutilização – DRU Gerencia de Reutilização – GRU (evolução) Analise de Decisão e Resolução - ADR	AP 1.1, AP 2.1, AP 2.2, AP 3.1 e AP3.2
D – Parcialmente Definido	Verificação – VER Validação – VAL Projeto e Construção do Produto – PCP Integração do Produto- ITP Desenvolvimento de Requisitos - DRE	AP 1.1, AP 2.1, AP 2.2, AP 3.1 e AP3.2
E – Largamente Definido	Gerencia de Projetos – GPR (evolução) Gerencia de Reutilização – DRU Gerencia de Recursos Humanos – GRH Definição do Processo Organizacional – DFP avaliação e Melhoria do Processo Organizacional - AMP	AP 1.1 AP 2.1 AP 2.2 AP 3.1 AP 3.2
F - Gerenciado	Medição – MED Garantia de Qualidade – GQA Gerencia de Configuração – GCO Aquisição - AQU	AP 1.1 AP 2.1 AP 2.2
G – Parcialmente Gerenciado (mais baixo)	Gerencia de Requisitos – GRE Gerencia de Projetos - GPR	AP 1.1 AP 2.1

Quadro 3.2 - Níveis do MPS.BR Processos e Capacidades
Fonte: MPS.BR, 2009.

O CMMI - Capability Maturity Model - é um modelo para avaliação da maturidade dos processos de software de uma organização e identificação das práticas-chave que são requeridas para aumentar a maturidade desses processos. Ele prevê cinco níveis de maturidade: 1 - Inicial; 2 - Repetível; 3 - Definido; 4 - Gerenciado ; 5 - Otimizado. (MCT, 2001).

O TMMi teve sua primeira versão baseada no SE-CMM e foi desenvolvido no Illinois Institute of Technology (Burnstein, 2002;TMMi, 2009;). Ele foi criado para tratar de forma específica o processo de teste de software. Baseado no SW-CMM, esse modelo também classifica os níveis de capacidade das empresas de executar os testes em 5 níveis, em que o nível 1 é o inicial e o 5 o nível mais alto. Os 5 níveis do TMMi são: 1 – Inicial; 2 – Fase de Definição; 3 – Integração; 4 – Gestão e Medições; 5 – Otimização.

O Quadro 3.3 mostra uma comparação entre as áreas-chave e os níveis de maturidade entre os modelos de melhoria TMMi e CMMI.

Nível TMMi	**Nível CMMI**	**Áreas Chave**
2	**2**	Gerência de requisitos, planejamento de projeto e gerência de configuração de software
3	**2**	Projeto de monitoramento e supervisão de SQA e metas de Controle de Qualidade.
3	**3**	Foco no processo Organizacional, Foco na organização, no processo de definição, e programas de treinamento.
4	**4**	Revisão por pares
4	**4**	Gerenciamento da qualidade de software e gerenciamento da quantidade de processos
5	**5**	Gerenciamento dos processos de mudança, gerenciamento de mudança de tecnologia e prevenção de defeitos

Quadro 3.3 - Áreas chave do modelo CMMI X Níveis de maturidade do TMMi
Fonte: MPT.BR, 2009.

O TMMi possui 13 (treze) objetivos de maturidade nos seus 5 (cinco) níveis. Para cada objetivo de maturidade, subobjetivos mais concretos são definidos para que o objetivo possa ser sustentado adequadamente. Atividades e tarefas são ações específicas para melhorar a capacidade de teste e as responsabilidades por sua execução são passadas para gerentes, desenvolvedores/testadores e usuários/clientes. Pode-se garantir que, embora exista alguma semelhança entre os dois, quanto mais se estuda o TMMi, mais longe ele fica do modelo CMMI. Basta lembrar que o TMMi não exige evidências do cumprimento de nenhuma das suas

práticas. De qualquer forma, fica a evidência de que é possível traçar uma linha de equivalência entre os modelos de maturidade de teste de software e de desenvolvimento de software. O MPT.BR foi criado para ser um modelo para avaliação da maturidade das áreas de teste de software compatível, mas não idêntico ao modelo MPS.BR, sendo que ele é totalmente voltado para a área de teste de software. Dessa forma, a empresa que implementar o modelo MPS poderá, com pouco esforço, implementar o modelo MPT. No entanto, a implantação do MPT não depende do uso do MPS pela empresa (MPT.BR, 2009).

Nível 0	Não tem Modelo de teste Implantado
Nível 1	Gerência de Projetos de Teste – GPT
Nível 2	Gerência de Requisitos de Teste – GRT
Nível 3	Aquisição – AQU (opcional)
Nível 4	Gerência de Recursos Humanos – GRH
Nível 5	Desenvolvimento de Requisitos – DRE
Nível 6	Análise de Decisão e Resolução – ADR
Nível 7	Análise de Causas e Resolução de Problemas – ACR

Quadro 3.4 – Níveis Áreas de processo do modelo MPT.BR
Fonte: MPT.BR, 2009.

O Guia de implementação do MPT.BR está subdividido em sete níveis equivalentes aos níveis de maturidade do MPS.BR (Quadro 3.4). A exceção é o nível 1, que contempla apenas a área de processo de Gerência de Projetos. O objetivo é atender áreas de teste muito pequenas. O nível 2 foi definido com as áreas de processo Gerência de Projetos e Gerência de Requisitos, de forma a permitir que a empresa possa, também, começar a implementação neste nível 2. Para facilitar o entendimento, e apenas neste caso, a área de Gerência de Projetos existe nos dois níveis, 1 e 2. A área de gerência de projeto de teste de software do nível 1 contempla menos resultados esperados do que a mesma área no nível 2. Conforme ALATS, 2009 o quadro 3.5 compara os níveis entre MPT, MPS e CMMI :

MPT	MPS	CMMI
1) Nível 0;	Sem correspondência	Nível 0
2) Nível 1;	Sem correspondência	Sem correspondência
3) Nível 2;	Nível G	Sem correspondência
4) Nível 3;	Nível F	Nível 2
5) Nível 4;	Nível E	Sem correspondência
6) Nível 5;	Nível D	Sem correspondência
7) Nível 6;	Nível C	Nível 3
8) Nível 7	Nível A e B	Nível 4

Quadro 3.5 - Comparativo dos níveis de maturidade MPT X MPS X CMMI
Fonte: MPT.BR, 2009.

3.6 Conclusão

Conforme Rios e Moreira Filho (2002), existe, atualmente, um grande esforço no sentido de integrar os modelos de maturidade para capacitação de software com os modelos de maturidade de testes. Os modelos já existem, a dificuldade das empresas é tomar a decisão de implantá-los e perseverar na tarefa. Ao utilizar como referência os modelos MPS.BR, CMMI, TMM e MPT.BR, considera-se que as semelhanças desses modelos facilitarão a compreensão da importância da utilização dos conceitos e das melhores práticas propostas por cada um deles.

4 PROCESSO PARA TESTES DE SISTEMAS DE SOFTWARE CRÍTICOS

4.1 Introdução

Baseando-se nos conceitos de testes de software, este trabalho apresenta uma abordagem para apoiar testes em softwares para Sistemas Críticos. Define um processo para modelar as atividades de teste que devem ser realizadas ao longo do projeto e um mapeamento entre este processo e o processo de desenvolvimento, indicando em quais pontos do processo de desenvolvimento as atividades de teste devem ser inseridas.

A Seção 4.2 apresenta o processo de teste definido, as normas nas quais ele se baseia e descreve sua dinâmica evolutiva, acompanhando o desenvolvimento do sistema. A Seção 4.3 apresenta a modelagem detalhada de cada atividade do Processo de Testes, descreve o objetivo de cada uma delas, critérios de entrada e de saída, os responsáveis e as tarefas de cada atividade. Na Seção 4.4, a título de exemplo, são detalhados alguns critérios de entrada e saída do modelo, na Seção 4.5, são exibidas algumas características de qualidade necessárias aos Sistemas Críticos e a Seção 4.6 apresenta as considerações finais deste capitulo.

4.2 O Processo de Teste Para Softwares Críticos

O processo de testes de software é composto de um conjunto de atividades e tarefas relacionadas ao planejamento e à execução dos testes, de acordo com o planejamento realizado. Esse processo trata os testes de artefatos produzidos ao longo de todo o ciclo de vida de um projeto de desenvolvimento de software.

Esquematicamente a dinâmica do processo de testes proposto para sistemas críticos é apresentado na figura 4.1:

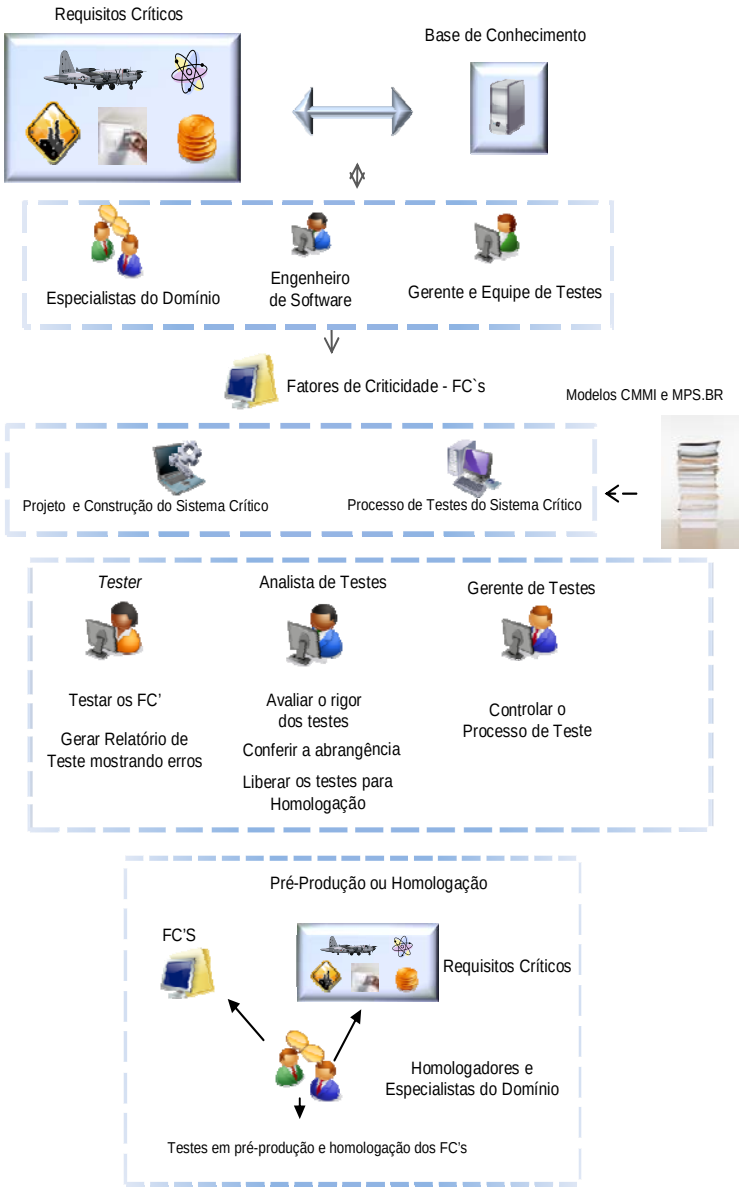


Figura 4.1 - Dinâmica do processo proposto para testes de sistemas críticos

Na Figura 4.1, é mostrado o estreito relacionamento entre as diversas fases de desenvolvimento do sistema com as atividades de teste que devem ser planejadas, executadas e avaliadas durante todo o ciclo de desenvolvimento. As atividades a serem realizadas por atores especialistas devem ser claras e objetivas. Cada um deve saber exatamente o que se espera dele.

Na primeira atividade do desenvolvimento do sistema crítico, a Análise de Segurança, os especialistas do domínio, os engenheiros de software, a equipe de testes e os usuários responsáveis pelo sistema vão levantar os requisitos críticos, inteirando-se de todas as características e particularidades do sistema crítico a ser desenvolvido. A disponibilidade de uma base de conhecimento em que todas as

informações disponíveis sobre sistemas semelhantes possam ser acessadas rapidamente é altamente recomendável.

Se essa base não existir, deve ser iniciada a construção da mesma. Sistemas Críticos devem começar a ser desenvolvidos utilizando conhecimento. Sabe-se que as equipes mudam, pessoas saem da equipe e levam parte do conhecimento, pessoas novas chegam e tem de aprender rápido para se nivelarem àquelas que já estão produzindo e não tem tempo para ensinar. A base de conhecimento possibilita a consulta e a atualização de informações do sistema todo o tempo. Quando se trata de sistemas críticos, padrões de programação são úteis, para facilitar a compreensão de artefatos, e aumentam a segurança dos artefatos de missão crítica. (uma sugestão de padrão consta do Anexo 3).

O objetivo da Análise de Segurança é identificar os perigos ou ameaças a que o Sistema Crítico está exposto e os fatores de criticidade, que são consequência deles, referidos neste trabalho como FC's. As providências para a mitigação desses perigos devem ser definidas nessa fase. (Uma sugestão de formulário para o levantamento inicial usado no estudo de caso apresentado neste trabalho consta do Anexo 2).

O desenvolvimento do sistema, projeto físico, construção do sistema e o processo de testes do sistema crítico devem ser feitos paralelamente. A fase de construção deve ser focada em manter os perigos sob controle. O Processo de Testes executa os testes adequados a cada fase, atendendo aos procedimentos definidos na atividade de análise de segurança. As atividades de construção e o processo de testes apóiam-se nos modelos de melhoria de qualidade e de melhoria de processo de testes, utilizando o que desses modelos se aplica a um acréscimo de qualidade, produtividade e otimização de custos.

Após o processo de teste ter sido definido, ele deve ser executado. Isso deve acontecer desde o início do desenvolvimento do sistema sob a responsabilidade de três atores principais: o testador ou *tester,* o analista de testes e o gerente de testes. A equipe de testes, sua composição, sua especialidade e experiência dependem também dos FC’s sistema.

O *tester* é responsável por executar os testes seguindo os casos de testes construídos pelo analista de teste. Para desempenhar sua função, o *teste*r não precisa necessariamente conhecer as regras de negócio do sistema que está sendo testado. O seu trabalho consiste, grosso modo, na execução de um programa ou

rotina, seguindo um caso de teste, informando os dados de entrada e verificando se os dados de saída são iguais aos resultados esperados. O *tester* gera os relatórios de ocorrências de testes. Se o teste for correto, o relatório deve ser atualizado com essa informação, o programa ou rotina é liberado para o próximo teste e as evidências de testes devem ser arquivadas cronologicamente em bases de fácil identificação e acesso.

Se os resultados forem diferentes dos esperados, o *tester* deve encaminhar o relatório de ocorrência para a equipe de desenvolvimento ou manutenção, para que o programa ou rotina seja corrigido e retorne ao ambiente de teste para ser novamente testado. Esse é um ciclo que só termina quando o teste gera resultados iguais aos resultados esperados.

O analista de testes tem a atribuição de confeccionar os casos de testes e projetar as bases de testes no ambiente de teste, para conter dados que possibilitem executar todos os testes necessários, considerando situações corretas e situações de erro. Em se tratando de sistemas críticos, todos os FC’s devem ser contemplados nos casos de testes. As bases, ao serem geradas, devem ser mascaradas para atender ao requisito de confidencialidade, caso este requisito exista.

Para executar testes de situações de extremo risco ligadas aos FC’s, devem ser incluídos nas bases dados que permitam a simulação de situações impossíveis de serem testadas, devidos aos riscos que envolvem a ocorrência real do evento a ser testado. O analista de teste é o ator que, no caso do nosso processo, revisa o trabalho do *tester*, utilizando os relatórios de ocorrência e libera os artefatos como já testados.

O Gerente de Testes atua a nível gerencial, verificando se os casos de testes contemplaram os FC’s do sistema, se os testes estão sendo executados conforme o cronograma e se o índice de erros está acima do esperado, caso existam métricas prévias. Uma tarefa importante do gerente de testes é atuar como facilitador, cuidando para que os ambientes de teste estejam sempre disponíveis e solucionando dificuldades e situações imprevistas que dificultam o trabalho da equipe de testes.

Assim que os artefatos são liberados pela equipe de testes, eles devem ser transferidos para o ambiente de pré-produção, também chamado de ambiente de homologação, nesse ambiente os artefatos serão agora testados pelos homologadores e especialistas do domínio. Nessa fase os testes são mais rigorosos

e entram em casos de testes mais completos e abrangentes, mais focados no FC's do sistema crítico. Essa é a ultima fase antes do sistema ser transferido para o ambiente de produção. No caso de Sistema Crítico, a transferência deve ser liberada formalmente pelos usuários definidos como responsáveis pela homologação do sistema.

Após essa visão de como os dois processos (o de desenvolvimento e o de teste) interagem durante a fase de construção, é importante compreender o funcionamento da atividade de teste e de suas fases. Apesar do modelo propor que as atividades de testes devem ser executadas gradativamente, durante a construção do sistema crítico, existe uma situação em que isso não pode ser feito. É quando o sistema não vai ser desenvolvido e somente implantado. Ou seja, o sistema já existe e será implantado somente em uma nova instalação ou planta. Sendo um sistema crítico, apesar de pronto e funcionando, ele terá que ser implantado ou instanciado e, para isso, deve ser testado. Nesse caso é a equipe de testes que fornece o apoio necessário à equipe de implantação para atingir o completo entendimento do funcionamento do sistema. As dúvidas podem ser resolvidas graças aos testes executados.

O Processo de Teste para sistemas críticos proposto deve ser diferente do processo de testes de um sistema não crítico, principalmente no cuidado e rigor que as equipes de testes e os homologadores devem ter ao executar as atividades e tarefas planejadas no ciclo de teste do sistema.

Todos os atores participantes das atividades de definição, execução, conferência e homologação devem ter conhecimento e devem ter sido sensibilizados da importância dos requisitos críticos funcionais e não funcionais, dos perigos que afetam o sistema e dos FC's a que o sistema está exposto. A atitude de alerta constante não deve ser somente uma postura inicial, mas uma política sistemática de orientação constante aplicada a toda equipe.

Propõe-se um processo que tem uma linha mestra a ser seguida que pode ser adaptada para as particularidades de cada sistema. Nesse processo estão relacionadas as atividades principais, que não devem ser simplificadas ou abreviadas, por isso o apoio dos níveis hierárquicos superiores às equipes de teste é imprescindível.

4.3 Modelagem Detalhada do Processo de Teste para Sistemas Críticos

O Processo de testes para Sistemas Críticos definido envolve a execução de quatro macroatividades conforme ilustrado na Figura 4.3 (o significado dos elementos gráficos utilizados é apresentado no Anexo 1):

- **Análise de Segurança do Software:** o objetivo desta macroatividade é identificar todas as ameaças da aplicação crítica. Devem ser identificados: os estados inseguros, as sequências de falhas que originam os estados inseguros, as causas das falhas, as possíveis conseqüências e as probabilidades das falhas acontecerem. Com esses dados é possível classificar os fatores críticos, os FC's, de cada ameaça ou estado inseguro. Os FC's serão essenciais para definição das providências necessárias para minimizar as ameaças. Finalmente, deverá ser feita uma avaliação de aceitabilidade;
- **Gerar o Processo de Testes:** o objetivo desta macroatividade é instanciar o processo de testes para o sistema crítico em questão. Os artefatos de saída da macroatividade anterior: as ameaças, os estados inseguros e os FC`s serão o ponto de partida para a geração do processo de teste adequado. As políticas de criação das bases de testes, padrões de nomes de arquivos e métricas de quantidade de casos de testes também derivam dos FC's. Se o sistema estiver em fase de construção, é a hora de definir os nomes de programas, nomes de arquivos, variáveis globais e outros padrões para executar e documentar os testes, definir os documentos a serem gerados e os procedimentos para resolução de diferenças entre os dados previstos e reais nos testes. As métricas mais adequadas ao sistema e que vão ser acompanhadas. Os responsáveis pela homologação dos testes também devem ser definidos nesse momento;
- **Planejar Testes:** o objetivo desta macroatividade é definir o plano de testes para o projeto, identificando os tipos de testes e os artefatos que devem ser testados, sempre usando os fatores de criticidade das ameaças vinculados ao artefato identificado. Essa atividade deverá ser coordenada pelo Gerente de Teste e pelo Gerente de Projeto, após terem sido selecionados os artefatos responsáveis pela minimização das ameaças identificadas na primeira macroatividade. Essas ameaças são fatores que caracterizam o sistema como um ‘Sistema Crítico’. Uma vez identificados esses artefatos: programas, rotinas, subrotinas, arquivos, equipamentos, softwares de transmissão, sistemas operacionais, instalações, e

outros, dependendo do domínio, eles devem ser eleitos para fazer parte de um conjunto de artefatos sujeitos a Testes de Missão Crítica. Para eles, deverá ser definida uma política de testes mais robusta, na qual serão aceitas redundâncias de tarefas de testes, verificação e acompanhamento. Aconselha-se, em alguns casos, políticas de testes executadas por empresas externas com o objetivo de manter uma documentação atualizada, prover maior exatidão e uma realização mais completa da atividade de testes. (No Anexo 5 há um exemplo de plano de teste utilizado no banco ABC);

- **Executar Testes:** o objetivo desta macroatividade é testar os artefatos selecionados na macroatividade anterior, de acordo com o plano de testes. Dependendo do grau de criticidade do sistema, além dos testes definidos e executados para o artefato, será necessária a verificação do artefato. A verificação deve ser executada através de, pelo menos, uma revisão por pares além do teste, de acordo com o que foi planejado. A seguir cada uma das macroatividades será especificada detalhadamente.

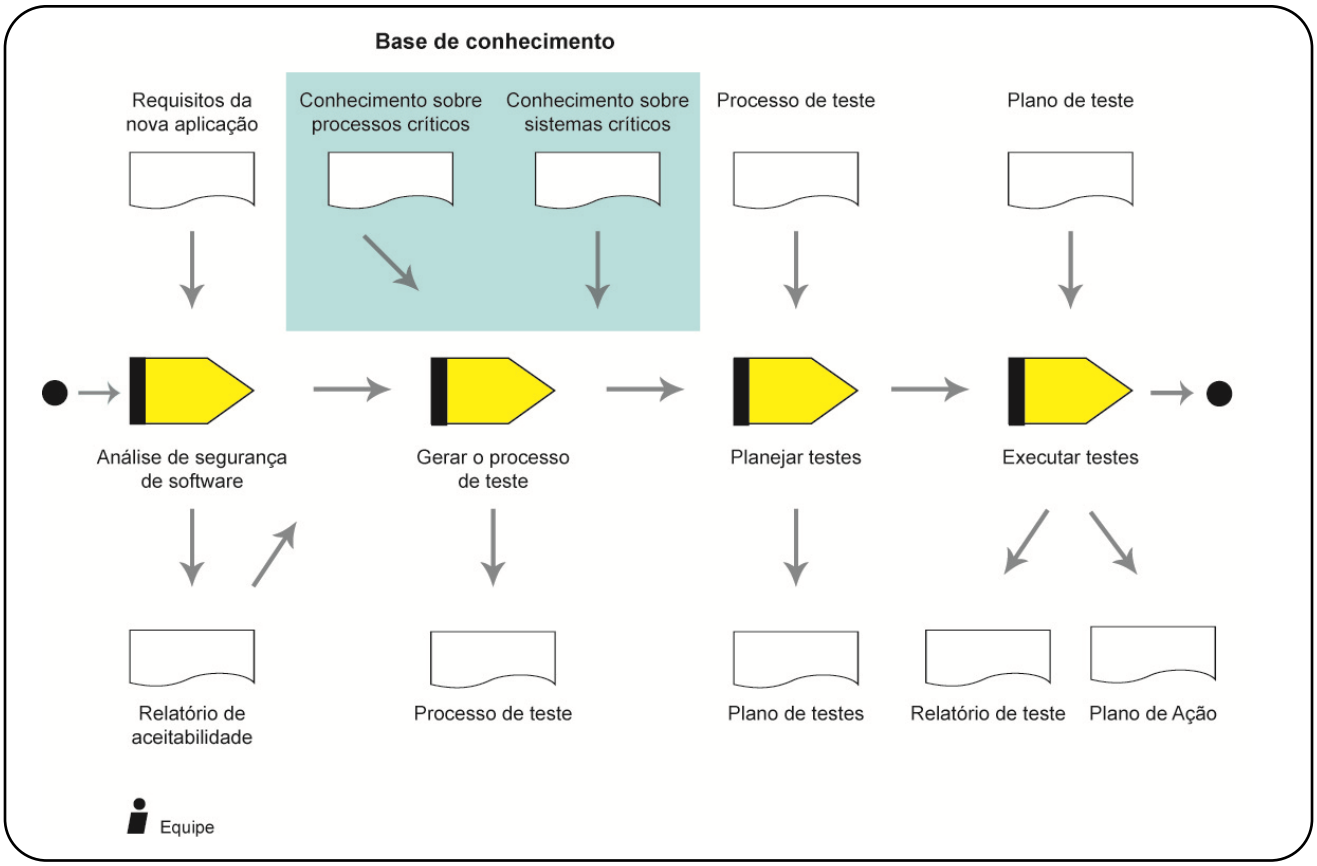


Figura 4.2 – Macroatividades do Processo de Testes

A primeira macroatividade do processo, Análise de Segurança do Software, pode ser dividida em quatro atividades:

(i) Identificação de estados inseguros;

(ii) Identificação de sequência de falhas;

(iii) Determinação dos fatores críticos dos estados inseguros; e

(iv) Avaliação da Aceitabilidade.

A Figura 4.3 ilustra as atividades que compõem a Análise de Segurança da Aplicação Crítica:

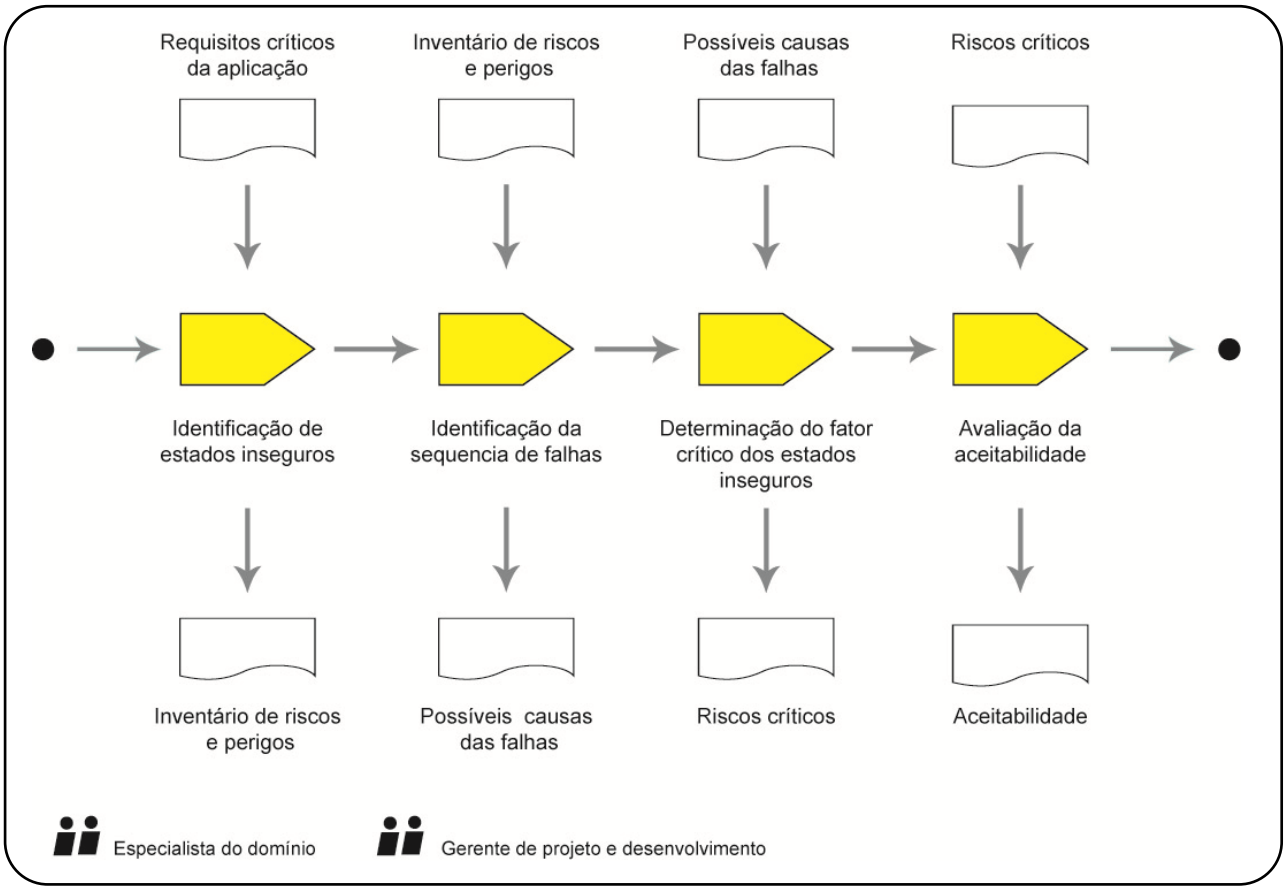


Figura 4.3 - Detalhamento da Macroatividade Análise de Segurança da Aplicação Crítica.

(i) Identificação dos estados inseguros:

Descrição: o objetivo dessa atividade é identificar os Estados Inseguros e Perigos, tendo como base as características do sistema (LEVESON e HARVEY; 1983; LEVESON, 1991; MOURA, 1996). Divide-se em três tarefas: Análise Preliminar de Insegurança (*Preliminary Hazards Analysis*), Análise de Insegurança

de Subsistemas (*Subsystem Hazards Analysis*) e Análise de Insegurança de Sistema (*System Hazards Analysis*).

Artefatos Requeridos: lista preliminar de requisitos críticos da aplicação identificados.

Artefatos Produzidos: inventário de riscos e perigos.

Critérios de Entrada: não há.

Critérios de Saída: perigos identificados e classificados conforme Critério de Classificação de Perigos.

Responsável: gerente de projeto e especialistas do domínio.

Tarefas:

- Fazer a Análise Preliminar de Insegurança do sistema que determinará os primeiros indícios para discriminar o sistema como crítico, conforme os critérios propostos no padrão britânico Int Def Stan 00-56 (MINISTRY OF DEFENSE, 2000);
- Consultar a base de conhecimento para acessar os perigos e riscos de sistemas semelhantes ou relacionados (preparação para o levantamento);
- Envolver o analista de testes nos primeiros levantamentos dos requisitos com o analista de requisitos\negócio\sistemas;
- Documentar o levantamento, se possível, gravando as reuniões. O analista de testes deve revisar e testar os requisitos críticos após sua documentação;
- Propor alternativas de controle para os perigos e riscos identificados;
- Completar a Base de Conhecimento com os perigos listados no inventário de perigos e com aqueles que não estejam na base, incluindo as descrições detalhadamente; e
- Identificar os dados recebidos ou enviados aos subsistemas que se comunicam com o sistema crítico. Eles serão selecionados para serem submetidos a teste de missão crítica.

(ii) Identificar seqüência de falhas:

Descrição: o objetivo dessa atividade é identificar as falhas que colocam o sistema em situação de perigo já identificadas. A consulta às bases de conhecimento e bases históricas de sistemas pertencentes ao mesmo domínio é

necessária para conseguir essa identificação. Para cada possível falha identificada, deve ser construído um caso de uso, descrevendo detalhadamente o evento e os atores envolvidos. Esse caso de uso será a base para a construção dos casos de teste a que serão submetidos os artefatos relacionados a essa falha;

Artefatos Requeridos: inventário de riscos e perigos identificados no sistema crítico;

Artefatos Produzidos: lista de causas das falhas. Construir também os casos de uso descrevendo as possíveis falhas. Providências definidas para evitar as conseqüências de cada falha identificada;

Critérios de Entrada: falhas identificadas classificadas como FC's críticos;

Critérios de Saída: lista de possíveis falhas homologada por especialistas do domínio e cadastrada na base de conhecimento;

Responsável: gerente de projeto e especialistas do domínio;

Tarefas:

- Analisar detalhadamente cada falha possível e construir casos de uso completos, esgotando o máximo de caminhos identificados;
- Homologar as falhas inventariadas com os especialistas do domínio e principais usuários identificados;
- Atualizar a base de conhecimento com as informações identificadas e iniciar a construção dos casos de teste, focando as possíveis falhas e perigos.

(iii) Determinação dos fatores críticos dos estados inseguros:

Descrição: associar a cada estado inseguro ou perigo identificado um fator de Criticidade ‘FC’;

Artefatos Requeridos: lista de perigos identificados no sistema crítico, probabilidade de ocorrência de cada perigo, número de pessoas possivelmente afetadas e perda financeira provável;

Artefatos Produzidos: lista de fatores de criticidade, FC's de cada perigo;

Critérios de Entrada: casos de uso, detalhando cada possível falha;

Critérios de Saída: fatores críticos de cada perigo identificados;

Responsável: gerente de projeto, gerente de testes e especialistas do domínio;

Tarefas:

- Consultar a base de conhecimento e atualizá-la com os dados levantados;
- Classificar cada perigo por tipo de severidade (catastrófica, crítica, marginal ou desprezível);
- Classificar cada perigo, conforme probabilidade de ocorrência do evento (freqüente, provável, ocasional, remota ou improvável);
- Fazer o cruzamento das duas classificações anteriores, atribuindo a cada perigo, o FC de T1 a T4, conforme Quadro 2.2.

(iv) Avaliação da aceitabilidade:

Descrição: obter um acordo com os especialistas do domínio e principal usuário homologador, para cada Perigo identificado e quanto ao FC de cada perigo.

Artefatos Requeridos: lista de fatores de criticidade, lista de perigos identificados no sistema crítico, lista de possíveis falhas homologada por especialistas do domínio, fatores de criticidade definidos e relatório de providências definidas para cada falha identificada,

Artefatos Produzidos: contrato entre os usuários do sistema e equipe de testes representados pelo gerente de teste. Escopo dos testes a serem executados;

Critérios de Entrada: fatores de criticidade definidos**;**

Critérios de Saída: providências definidas homologadas;

Responsável: gerente de projeto, gerente de testes, especialistas do domínio e usuário responsável pelo sistema;

Tarefas:

- relacionar os perigos e os FC's a ele atribuídos;
- relacionar para cada um deles as providências levantadas para manter os perigos sobre controle;
- avaliar os Estados Inseguros não evitados, controlados ou recuperados completamente, para determinar se os mesmos realmente não representam uma ameaça à segurança do software, garantindo um nível aceitável de risco;

- encaminhar formalmente para os especialistas do domínio e para o usuário principal do sistema, solicitando homologação formal das providências.

A segunda macroatividade do processo, Gerar o Processo de Testes da Aplicação Crítica, pode ser dividida em três atividades:

(i) Compor equipe de testes;

(ii) Definir ambiente e laboratório de teste; e

(iii) Definir as estratégias e políticas de testes.

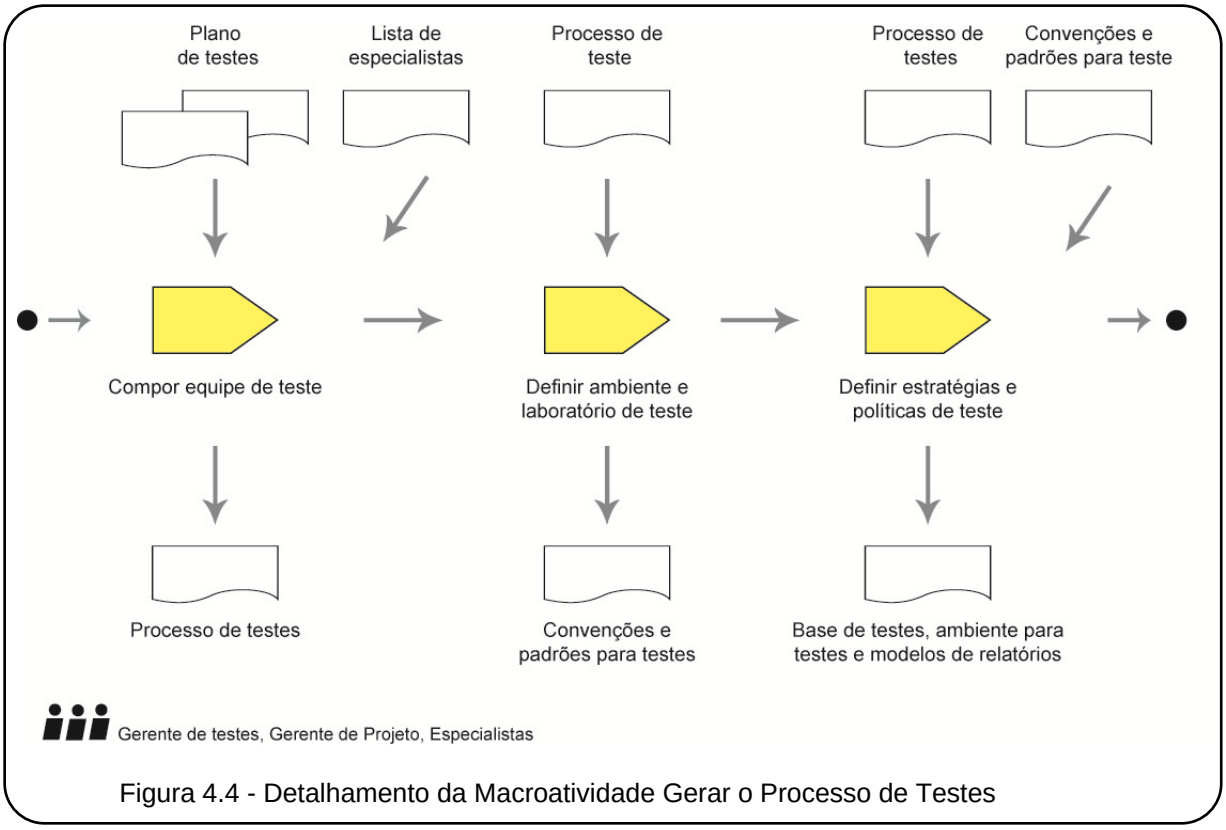


Figura 4.4 - Detalhamento da Macroatividade Gerar o Processo de Testes

(i) Compor equipe de testes:

Descrição: essa atividade tem como objetivo montar uma equipe que se identifique com a atividade de testar sistemas. A equipe deve conter pessoas que tenham experiência nas atividades de teste e pessoas com conhecimento no domínio do sistema. O responsável pela composição dessa equipe deve ter conhecimento dos perigos e dos FC's do sistema. Para atender ao quesito custo, as pessoas mais experientes devem ser alocadas em atividades de coordenação ou conferência em níveis mais altos, caso apresentem o perfil adequado. É muito importante que a equipe seja acompanhada de perto e esteja altamente motivada

em encontrar defeitos e solucioná-los. Nesta fase será definido o perfil de cada cargo do time de testes como, por exemplo, gerente de testes, líder de testes, engenheiro de testes e analista de testes. Cada um desses cargos deve exigir requisitos mínimos de experiência para ser preenchido;

Artefatos Requeridos: lista de candidatos ou profissionais pré-selecionados com a descrição de suas experiências anteriores em testes e no domínio do sistema crítico e relatórios de avaliação de perfil;

Artefatos Produzidos: lista de pessoas selecionadas, relacionando a atribuição principal e secundária, se houver, descrição detalhada das responsabilidades de cada um e sua subordinação, definição do regime de trabalho e datas de férias previstas, inventário de equipamentos necessários, ferramentas de software de apoio, mesas, cadeiras, telefone, alimentação, caso necessário, políticas de reembolso de transporte emergencial e outros itens necessários para que seja alcançada a condição de trabalho ideal para os técnicos;

Critérios de Entrada: descrição em detalhes de todas as qualificações que o profissional deverá possuir obrigatoriamente e outras qualificações desejáveis, como: certificações, anos de experiência no cargo, facilidade para trabalhar em equipe, ferramentas de automação, etc;

Critérios de Saída: equipe definida conforme a demanda de mão-de-obra prevista, considerando os FC's do Sistema Crítico, as redundâncias de testes definidas e o fator de segurança que será utilizado nos testes do sistema crítico;

Responsável: gerente de projeto, gerente de testes e especialistas do domínio.

(ii) Definir ambiente e laboratório de teste:

Descrição: essa atividade tem como objetivo definir um ambiente de desenvolvimento que possibilite a execução dos testes dos artefatos de forma controlada, repetitiva e abrangente.

Artefatos Produzidos: diagrama dos ambientes definidos. Nome de bibliotecas de artefatos, *books*, programas, telas, DLL’s, bibliotecas de documentação de evidências de teste definidas. Definição de políticas de autorização de acesso às bases de teste e homologação, definição de rotinas de

recuperação e preservação das bases durante a execução dos casos de teste. Horários de restrição e recuperação;

Critérios de Entrada: não há;

Critérios de Saída: definição do processo de teste iniciado com o ambiente de testes a ser utilizado;

Responsável: gerente de projeto, gerente de testes e especialistas do domínio.

(iii) Definir as Estratégias e Políticas de Testes:

Descrição: essa atividade tem como objetivo definir as estratégias e políticas de testes que serão utilizadas por toda a equipe de testes. Essa definição alinha a criação dos casos de teste, a execução dos testes e a geração dos relatórios de ocorrência de erro ou de resultado igual ao esperado. Os casos de testes são muitos e podem ser executados por pessoas diferentes. A definição formal das estratégias e políticas pretende reduzir a execução dos processos de várias formas diferentes.

Artefatos Produzidos: estratégias e políticas de testes definidas formalmente e documentada na base de conhecimento para ser acessada por toda a equipe de testes;

Critérios de Entrada: definição do processo de teste iniciado com o ambiente de testes a ser utilizado;

Critérios de Saída: processo de teste iniciado com as estratégias e políticas de testes definidas.

Responsável: gerente de projeto, gerente de testes e especialistas do domínio.

Como mostra a figura 4.5, abaixo, a macroatividade Planejar Testes dos Artefatos pode ser dividida em cinco atividades:

(i) Selecionar Métodos e Ferramentas;

(ii) Identificar Critérios e Tipos de Teste;

(iii) Identificar Responsáveis;

(iv) Definir Destinatários dos Relatórios de Testes; e

(v) Adaptar Modelo dos Relatórios de Testes.

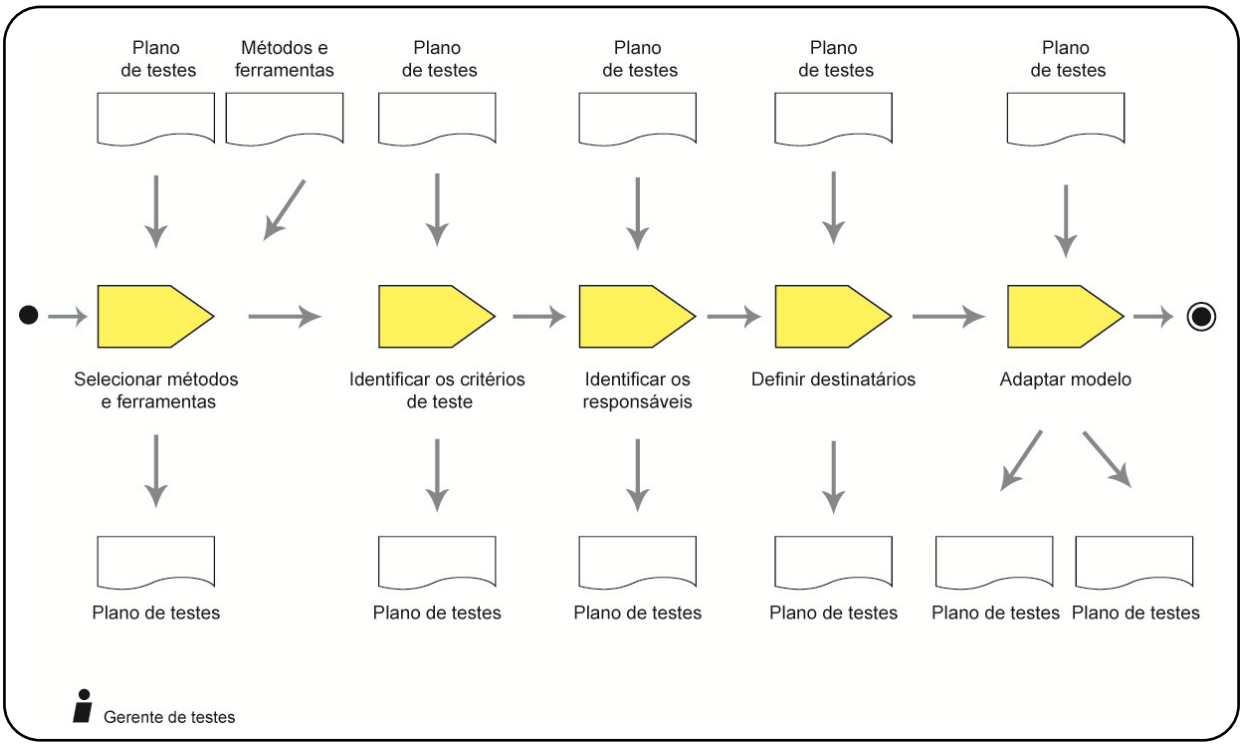


Figura 4.5 - Detalhamento da Macroatividade Planejar Testes dos Artefatos

(i) Selecionar Métodos e Ferramentas:

Descrição: essa atividade consiste em identificar, detalhar e viabilizar as ferramentas que serão usadas para testar os artefatos identificados e selecionados para serem testados;

Artefatos Requeridos: lista de métodos e ferramentas disponíveis, lista de requisitos identificados, matriz de rastreabilidade de requisitos X artefatos de software e, principalmente, lista de FC's X artefatos afetados;

Artefatos Produzidos: plano de testes preenchido gradualmente;

Critérios de Entrada: seleção dos artefatos para testes concluída;

Critérios de Saída: métodos e ferramentas a serem utilizados selecionados;

Responsável: gerente de testes.

(ii) Identificar Critérios e Tipos de Testes:

Descrição: essa atividade consiste em selecionar os critérios que serão usados para testar o artefato. Os tipos de testes a serem executados são definidos para garantir que o artefato funcione de acordo com os requisitos estabelecidos para o mesmo. Os artefatos que tratam os perigos com FC igual a T3 ou T4, classificados

como artefatos de missão crítica, devem ser submetidos a vários tipos de teste e ter os casos de testes mais detalhados. Deve ser perseguida, sempre, a maior cobertura do código possível;

Artefatos Requeridos: lista de perigos identificados, artefatos que controlam os perigos, plano de testes iniciado;

Artefatos Produzidos: planos de testes;

Critérios de Entrada: seleção dos artefatos para testes concluída;

Critérios de Saída: critérios para testes do artefato identificados;

Responsável: gerente de testes, analistas de testes.

(iii) Identificar Responsáveis:

Descrição: essa atividade consiste em definir a pessoa que será responsável pelos testes dos artefatos, bem como aquelas que participarão da atividade de teste propriamente dita;

Artefatos Requeridos: lista de pessoas que participam do projeto, lista de artefatos selecionados para teste;

Artefatos Produzidos: lista de artefatos a serem testados com os responsáveis pelo teste e com a homologação identificados;

Critérios de Entrada: seleção dos artefatos para testes concluída;

Critérios de Saída: responsável e participantes identificados;

Responsável: gerente de testes.

(iv) Definir Destinatários dos Relatórios de Testes:

Descrição: essa atividade consiste em identificar as pessoas que, ao final dos testes, receberão os relatórios de testes dos artefatos;

Artefatos Requeridos: lista de pessoas que participam do projeto;

Artefatos Produzidos: lista de artefatos a serem testados com os destinatários identificados;

Critérios de Entrada: seleção dos artefatos para testes concluída;

Critérios de Saída: destinatários dos relatórios identificados e informados de sua atribuição;

Responsável: gerente de testes.

(v) Adaptar Modelo dos Relatórios de Testes:

Descrição: essa atividade consiste em adaptar o modelo dos laudos de teste e/ou dos relatórios de testes que serão produzidos durante os testes dos artefatos, de acordo com os critérios selecionados. O objetivo dessa adaptação é torná-los mais adequados ao projeto. É necessário definir maneiras adequadas de arquivar as evidências de teste e definir maneiras práticas dos resultados de testes serem homologados pelos homologadores definidos; (SANDEEP MAHER, SR. TEST MANAGER, 2009).

Artefatos Requeridos: modelo de relatório de teste;

Artefatos Produzidos: modelo de relatório de teste adaptados ao sistema;

Critérios de Entrada: critérios para teste do artefato identificados;

Critérios de Saída: modelo dos relatórios disponibilizados na base de conhecimento aprovados pelo gerente de testes. Arquivamento das evidências de teste produzidas definida. Métodos de homologação definidos e contratados com os usuários homologadores;

Responsável: gerente de testes, homologadores.

Ao longo do planejamento dos testes, o Plano de Testes, contendo as informações do planejamento, é produzido e atualizado. Esse plano é o principal guia para a execução dos testes. Ao longo do processo, os artefatos que foram selecionados para serem testados devem ser testados, conforme planejado. Para isso a macroatividade Executar os Testes deve ser realizada.

A execução dos testes planejados para um artefato deve ser executada, cumprindo o *script* dos casos de testes negativos e positivos, ou seja, testes com resultados corretos e incorretos. Todas as condições identificadas para minimizar os riscos e perigos devem ser testadas, revisadas e verificadas por um segundo testador, através de, pelo menos, uma revisão por pares além do teste, de acordo com o que foi planejado. Essa macroatividade se divide em quatro atividades, como mostra a Figura 4.7. São elas:

(i) Executar testes;

(ii) Conferir os resultados dos testes;

(iii) Enviar relatórios de testes aos destinatários; e

(iv) Analisar os resultados dos testes.

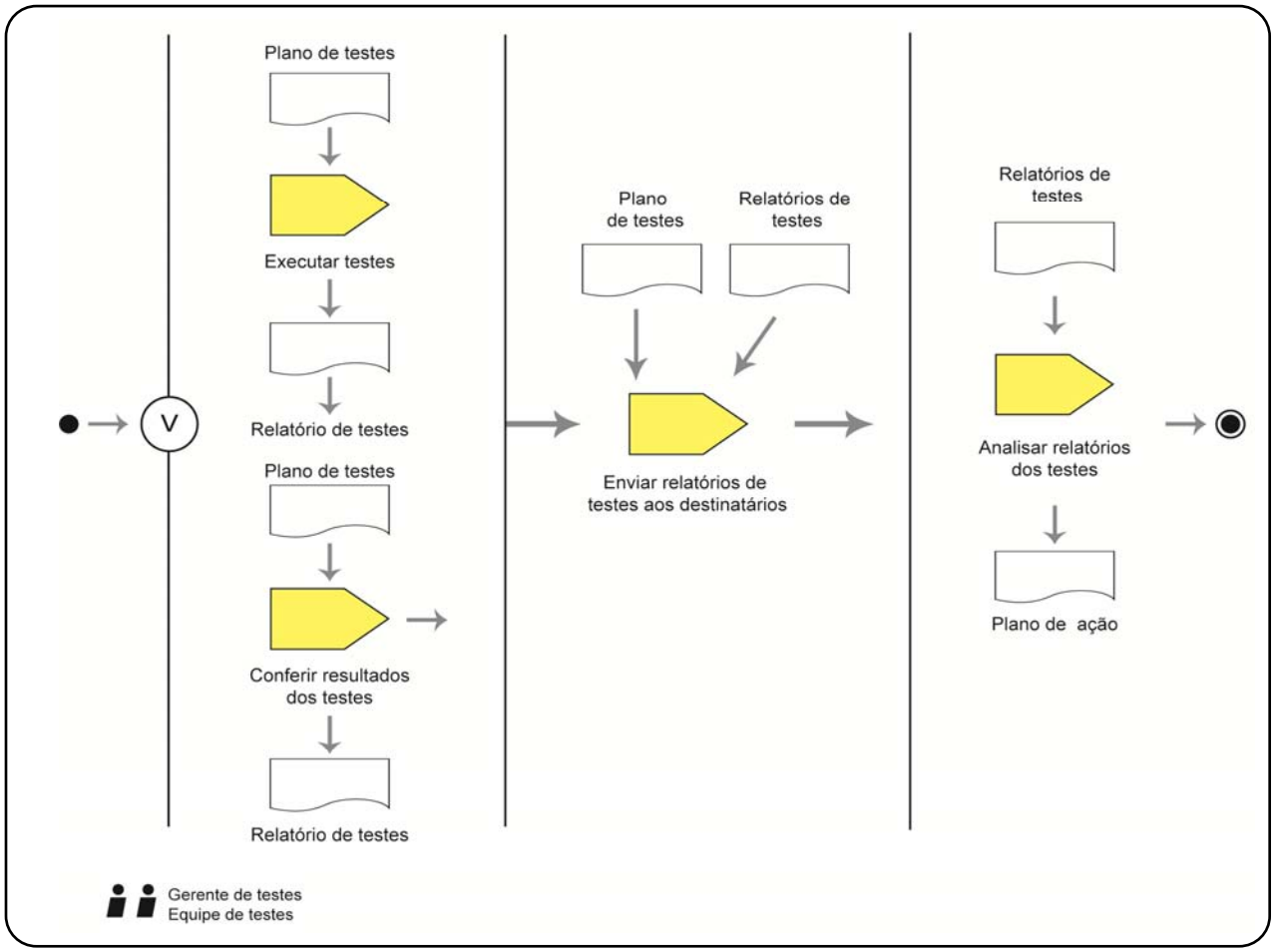


Figura 4.6 - Detalhamento da macroatividade Executar Testes

(i) Executar testes

Descrição: essa atividade consiste em executar os testes previstos, conforme descrito nos Casos de Testes construídos para o artefato. Pode ser realizado o teste unitário, o teste de integração do software, o teste do software, o teste do sistema, o teste de aceitação, dentre outros.

Artefatos Requeridos: casos de testes do artefato;

Artefatos Produzidos: evidências de testes e laudos de testes gerados pela equipe de teste;

Critérios de Entrada: codificação do artefato concluída;

Critérios de Saída: teste realizado no artefato;

Responsável: equipe de testes.

(ii) Conferir os Resultados dos Testes

Descrição: essa atividade consiste em conferir os resultados reais dos testes comparando-os com os resultados previstos no plano de teste. A conferência de resultados de testes deve ser feita em vários níveis. Quando é executado o teste unitário, o foco está no resultado gerado por um artefato. Quando outro tipo de teste está sendo conferido, (teste de integração, teste de regressão), o resultado pode ser muito mais impactante e pode exigir do testador um conhecimento muito profundo do negócio. Isso é um fator limitante da exatidão da conferência dos testes;

Artefatos Requeridos: teste do artefato executado, plano de teste;

Artefatos Produzidos: laudo de teste com parecer de liberação e promoção para ambiente pós-teste ou retorno à programação em caso de erro;

Critérios de Entrada: critérios para teste do artefato identificados;

Critérios de Saída: modelo dos relatórios adaptado;

Responsável: equipe de testes, equipe do projeto.

(iii) Enviar Relatórios de Testes aos Destinatários:

Descrição: essa atividade consiste em encaminhar os relatórios de testes produzidos nas duas atividades anteriores aos destinatários definidos no planejamento do testes do artefato em questão;

Artefatos Requeridos: plano de testes, relatórios de testes;

Artefatos Produzidos: homologação dos testes pelos usuários;

Critérios de Entrada: testes do artefato concluídos;

Critérios de Saída: relatórios de testes enviados aos destinatários. Em caso de erro, todos os dados para a repetição do erro devem ser informados no relatório. O testador deve tomar as providências necessárias para bloquear, nas bases, a alteração nos dados necessários para a repetição do erro. Isso acelera a identificação do erro no artefato, o acerto e novo teste de verificação do acerto;

Responsável: equipe do projeto e equipe de testes.

(iv) Analisar os Resultados dos Testes:

Descrição: essa atividade consiste em analisar os resultados dos testes executados com os resultados previstos e documentados nos casos de testes. A partir dessa análise, devem ser identificadas as ações necessárias para corrigir

resultados reais diferentes dos esperados. A equipe definida para esses trabalhos deve conhecer a fundo os requisitos do sistema e os casos de teste produzidos durante a fase de planejamento dos testes. As alterações necessárias identificadas pela equipe de teste durante a execução dos testes devem ser documentadas cuidadosamente antes de serem encaminhadas para os responsáveis definidos anteriormente;

Artefatos Requeridos: relatórios de testes, plano de teste do sistema e casos de teste;

Artefatos Produzidos: lista dos itens de ação;

Critérios de Entrada: testes do artefato concluídos;

Critérios de Saída: resultados dos testes analisados e itens de ação identificados;

Responsável: equipe de testes, gerente de teste e gerente do projeto.

Os diversos métodos de testes requerem um planejamento específico para cada artefato. Esse planejamento é uma tarefa complexa que tem sido investigada por diversos trabalhos (PERRY , 2000; CRAIG e JASKIEL, 2002; VILELA , 2004, KANER *et al.*, 2002; KUHN e REILLY, 2002; PHADKE, 2003) Por esse motivo, o planejamento específico para a aplicação dos métodos selecionados para o teste de um determinado artefato não será tratado neste trabalho.

Muito importante para a execução de um processo de testes é a definição das responsabilidades por cada atividade a ser executada. O Quadro 4.1 apresenta um resumo das responsabilidades descritas no processo de testes definido:

Atividades do Processo de Testes	Gerente de Testes	Equipe de Testes	Homologadores
Selecionar Artefatos para Testes	X		
Planejar os Testes do Artefato	X		
Realizar Testes		X	
Enviar Relatórios de Testes aos Destinatários		X	
Analisar os Resultados dos Testes		X	X
Homologar os Artefatos Testados			X

Quadro 4.1 - Responsáveis pelas atividades do processo de testes

4.4 Comparação com os modelos de Qualidade de Software

O processo definido neste trabalho se apóia na área de processo Verificação definida no modelo CMMI e no processo de Verificação definido no MPS.BR (MPS.BR, 2009). Utilizam-se os testes para materializar a Verificação. O quadro 4.2 relaciona as atividades definidas no processo com as práticas específicas de Verificação do CMMI. O Quadro 4.3 relaciona as atividades do processo definido para testes de sistemas críticos com os resultados esperados pelo processo de verificação definido no MPS.BR. O quadro 4.4 relaciona as atividades de gerência de projeto, GRPT, definidas no MPT.BR, com as atividades do modelo proposto.

Atividades do Processo / Prática Específica	Selecionar Artefatos para Testes	Planejar Testes do Artefato	Realizar Testes	Enviar Relatórios de Testes aos Destinatários	Analisar os Resultados do Teste
SP 1.1 Selecionar Produtos para Verificação	X	X			
SP 1.2 - Estabelecer o Ambiente para Verificação		X			
SP 1.3 – Estabelecer Procedimentos e Critérios para Verificação		X			
SP 2.1 - Preparar para Revisões por Pares		X			
SP 2.2 – Conduzir revisões por pares		X			
SP 2.3 – Analisar Dados das Revisões por Pares					X
SP 3.1 – Realizar a Verificação			X		
SP 3.2 - Analisar os Resultados da Verificação e Identificar Ações Corretivas				X	X

Quadro 4.2. Relacionamento entre as atividades do processo definido e a área de processo verificação do CMMI

Atividades do Processo / Prática Específica	Selecionar Artefatos para Teste	Planejar Teste do Artefato	Realizar Testes	Enviar Relatórios de Teste aos Destinatários	Analisar os Resultados dos Testes
VER 1. Uma estratégia de verificação é desenvolvida	X	X			
VER 2. Critérios para verificação de todos os produtos de trabalho requeridos são identificados	X	X			
VER 3. Atividades de verificação requeridas, incluindo revisões por pares, são executadas		X	X	X	X
VER 4. Resultados de atividades de verificação são analisados e disponibilizados para o cliente e outras partes envolvidas				X	X
VER 5. Defeitos são identificados e registrados e ações corretivas são determinadas					X

Quadro 4.3 - Relacionamento entre o processo definido e o processo de verificação do MPS.BR

O MPT.BR, 2009 determina que para implementar o nível 1 de maturidade do MPT.BR deve-se instalar a área de processo, Gerência de Projetos de Teste de Software (GRPT). O Quadro 4.4 relaciona os resultados esperados da GRPT com os Processos e Atividades do modelo de testes proposto.

Resultados Esperados da GRPT do MPT.BR	Processos e Atividades do Modelo
GPRT1 - O escopo do trabalho para o projeto é definido	
GPRT2 - As tarefas e os produtos de trabalho do projeto de teste são dimensionados utilizando métodos apropriados	Planejar Testes dos Artefatos – Selecionar Métodos e Ferramentas
GPRT3 - O modelo e as fases do ciclo de vida do projeto de teste são definidos	Gerar o Processo de Testes
GPRT4 - O esforço e o custo para a execução das tarefas e dos produtos de trabalho são estimados com base em dados históricos ou referências técnicas	Planejar Testes
GPRT5 - O orçamento e o cronograma do projeto de teste, incluindo marcos e/ou pontos de controle, são estabelecidos e mantidos	Planejar Testes
GPRT6 - Os riscos do projeto de teste são identificados e o seu impacto, probabilidade de ocorrência e prioridade de tratamento são determinados e documentados	Atividade de Análise de Segurança
GPRT7 - Os recursos humanos para o projeto são planejados considerando o perfil e o conhecimento necessários para executá-lo	Gerar o Processo de Testes – Compor Equipe de Testes
GPRT8 - As tarefas, os recursos e o ambiente de trabalho necessário para perfil e o conhecimento necessários para executar o projeto de teste são planejados	Gerar o Processo de Testes – Definir Ambiente e Laboratório de Testes
GPRT9 - Os dados relevantes do projeto de teste são identificados e planejados quanto à forma de coleta, armazenamento e distribuição. Um mecanismo é estabelecido para acessá-los, incluindo, se pertinente, questões de privacidade e segurança	Planejar Testes dos Artefatos – Selecionar Métodos e Ferramentas – Identificar Critérios de Testes- Definir Destinatários
GPRT10 - Planos para a execução do projeto de teste são estabelecidos e reunidos no Plano de Teste	Gerar o Processo de Testes – Definir Estratégias e Políticas de Testes
GPRT11 - A viabilidade de atingir as metas do projeto de teste, considerando as restrições e os recursos disponíveis, é avaliada. Se necessário, ajustes são realizados	Gerar o Processo de Testes – Definir Estratégias e Políticas de Testes
GPRT12 - O Plano de Teste é revisado com todos os interessados e o compromisso com ele é obtido	Gerar o Processo de Testes – Definir Estratégias e Políticas de Testes
GPRT13 - O progresso do projeto é monitorado com relação ao estabelecido no Plano de Teste e os resultados são documentados	Gerar o Processo de Testes – Definir Estratégias e Políticas de Testes
GPRT14 - O envolvimento das partes interessadas no projeto de teste é gerenciado	Gerar o Processo de Testes – Definir Estratégias e Políticas de Testes
GPRT15 - Revisões são realizadas em marcos do projeto e conforme estabelecido no planejamento	Gerar o Processo de Testes – Definir Estratégias e Políticas de Testes
GPRT16 - Registros de problemas identificados e o resultado da análise de questões pertinentes, incluindo dependências críticas, são estabelecidos e tratados com as partes interessadas	Executar Testes – Enviar Relatórios de Testes aos Destinatários
GPRT17 - Ações para corrigir desvios em relação ao planejado e para prevenir a repetição dos problemas identificados são estabelecidas, implementadas e acompanhadas até a sua conclusão	Executar Testes – Analisar Resultados dos Testes

Quadro 4.4 - Relacionamento Entre a Gerencia de Projetos GPRT do MPT.BR e o Modelo Proposto

Os modelos CMMI e MPS.BR não citam a atividade de homologação dos testes após a execução dos mesmos e nem a geração de evidências desses testes. Essa etapa, que no processo definido é mandatária para a liberação dos artefatos para transferência para ambientes de produção, é uma contribuição desse processo para a confiabilidade dos sistemas críticos.

4.5 Detalhamento dos Critérios de Entrada e Saída do Modelo

As atividades propostas no Modelo de Teste utilizam critérios de entrada e saída para orientar o fluxo de execução das atividades que dependem das atividades anteriores e, também, para permitir a definição e acompanhamento de atividades paralelas. Esses critérios devem ser construídos conforme o nível de criticidade dos requisitos para facilitar o acompanhamento e segundo o nível de sucesso alcançado durante a execução dos testes.

Estes critérios podem evoluir ao longo do tempo, conforme a organização ganha conhecimento e amadurece desenvolvendo softwares críticos. Os quadros a seguir mostram os critérios propostos neste trabalho. Eles estão preenchidos de acordo com o caso considerado no capítulo 6.

CRITÉRIO PARA LIBERAÇÃO DE CASOS DE TESTES						
Tipo de Artefato: **Código do Artefato:** **Função:**						
Critérios	**Avaliação**					
Caso de Teste Executado?	5	Sim X			0	Não
Resultado Real = Esperado ?	5	Sim X			0	Não
Evidência de teste gerada?	5	Sim X			0	Não
Nível de Complexidade	1	Alta	1	Média	5	Baixa X
Total	16					
Considerações para utilização deste critério: • Artefatos com pontuação acima de 15 devem ser selecionados.						

Quadro 4.5 - Critério para liberação de casos de testes

Critério de classificação de artefato de missão crítica						
Tipo de Artefato: programa rotina relatório procedimento operacional (marque x na frente) **Código do Artefato: Função:**						
Critérios	**Avaliação**					
Atualiza Bases?	5	Sim			1	Não
Controla Fator Crítico?	5	Sim			1	Não
Pertence a Rotina (24 x7) ?	5	Sim			1	Não
Nível de Complexidade	5	Alta X	2	Média	1	Baixa
Total	20					
Considerações para utilização deste critério: • Artefatos com pontuação acima de 10 devem ser selecionados.						

Quadro 4.6 - Critério de classificação de artefato de missão crítica

CRITÉRIO PARA CLASSIFICAÇÃO DE AMEAÇAS ELICITADAS											
Ameaça: Não liberação de empréstimo aprovado											
Critérios		**Probabilidade**									
Tipo de Severidade		**Frequente**		**Provável**		**Ocasional**		**Remota**		**Improvável**	
Risco de perda de Vidas humanas											
Até 1 vida		1	X	1		1		1		1	
De 1 ate 100 vidas		3		2		2		2		2	
Mais de 100 vidas humanas		5		5		5		3		3	
Perda financeira até 1000 reais		1	X	1		1		1		1	
Perda financeira entre 1001 e 100000 reais		3	X	3		3		2		2	
Perda financeira entre 100001 e 1M de reais		5		5		5		2		2	
Soma de Pontos			5								
Total	**Considerações para utilização deste critério:** • Ameaças com pontuação acima de 10 devem ser discriminadas como críticas • Soma de Pontos > ou = 5 + Freqüente o FC = T4 • Soma de Pontos > ou = 5 + Provável o FC = T3 • Soma de Pontos > ou = 5 + Ocasional o FC = T2 • Soma de Pontos > ou = 5 + Remota ou improvável FC = T1										

Quadro 4.7 - Critério para classificação de ameaças elicitadas

CRITÉRIO PARA MONTAGEM DA EQUIPE DE TESTES				
Projeto:				
Cargo: Analista de Testes **Nome do Candidato: Jose Antonio de Castro Data: 10/01/1998**				
Avaliação de Competências	**Avaliação**			
O candidato tem experiência comprovada em testes de sistemas?	5	Sim	0	Não
O candidato já trabalhou em desenvolvimento de sistemas críticos?	5	Sim	0	Não
O candidato já trabalhou em manutenção de sistemas críticos?	5	Sim	0	Não
O candidato já trabalhou em testes de sistemas críticos?	5	Sim	0	Não
O candidato possui algum tipo de especialização ou certificação?	5	Sim	0	Não
O candidato tem algum conhecimento do domínio do Sistema?	5	Sim	0	Não
O candidato conhece Linguagem de Modelagem?	5	Sim	0	Não
Soma da pontuação:				
Considerações para utilização deste critério: Candidatos com pontuação acima de 20 considerar aptos a serem entrevistados para seleção				

Quadro 4.8 - Critério para montagem da equipe de testes

4.6 Conclusão

Este capítulo apresentou uma abordagem para a execução de testes em softwares de Sistemas Críticos fundamentada nos conceitos de segurança e redundância de procedimentos presentes na literatura. Foi proposta a utilização de um processo que organiza as atividades de um processo de teste a serem executadas e a integração deste processo ao processo de desenvolvimento de software. Foi sugerido um conjunto com informações sobre características de qualidade e critérios para auxiliar na execução das atividades de planejamento, execução e homologação dos testes.

5 FERRAMENTA DE APOIO AO PROCESSO DE TESTE

O objetivo deste capítulo é apresentar a ferramenta SATS, seu modelo implementacional, e exibir as principais telas disponíveis na ferramenta, juntamente com um exemplo.

5.1 Introdução

A ferramenta SATS, Sistema de Apoio a Testes de Sistemas Críticos, apóia o planejamento, a execução e o controle de testes em sistemas de softwares, discriminados como críticos, como pode ser visto na figura 5.1. A ferramenta cria e utiliza uma base de dados em que serão documentadas as atividades de um processo de teste para sistemas críticos e, principalmente, “como” elas devem ser realizadas, conforme definido pela equipe de teste e por especialistas no domínio do sistema crítico.

Os casos de teste criados estão relacionados aos requisitos críticos e aos fatores de criticidade identificados na fase inicial de análise. A ferramenta permite a

rastreabilidade dos testes, desde a sua concepção, até a sua execução. Essa rastreabilidade é uma característica de qualidade. Este capítulo apresenta as funcionalidades da ferramenta SATS que apóiam o Modelo proposto no Capítulo 4.

A Figura 5.1 ilustra as atividades, as tarefas e os produtos do modelo de processo de testes:

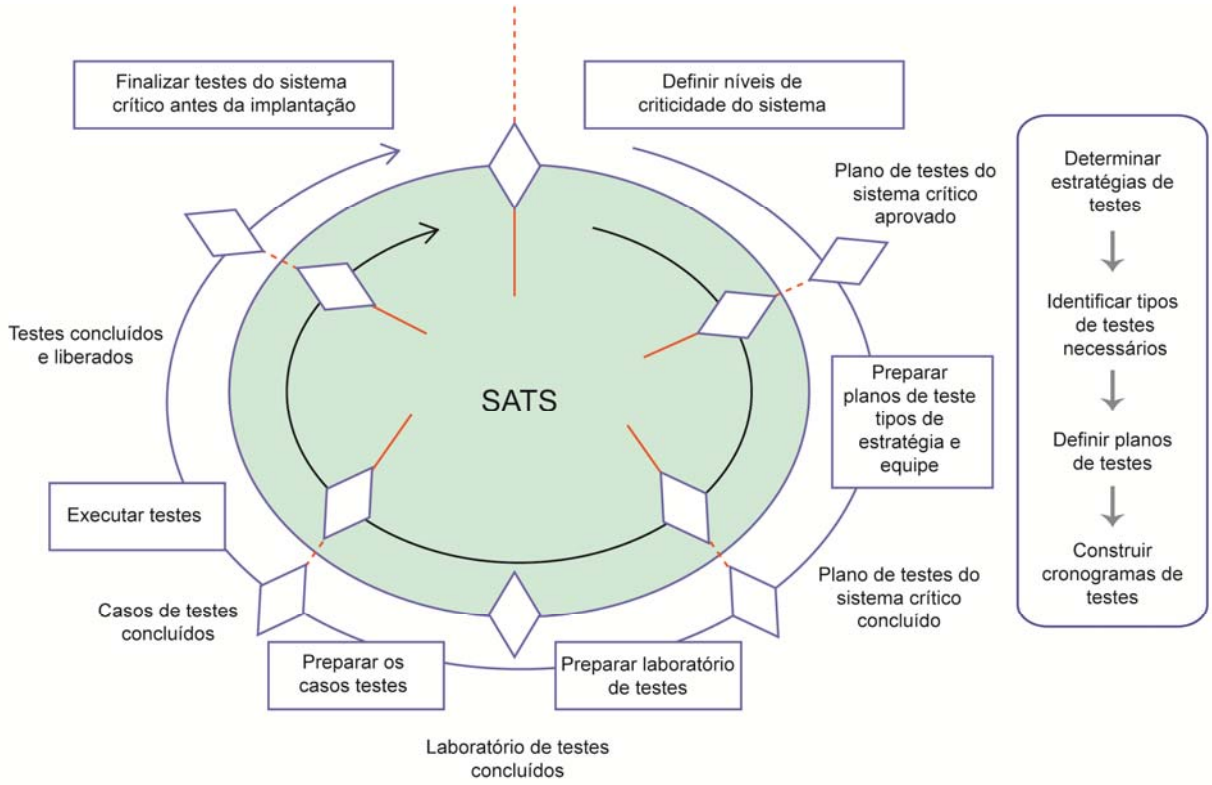


Figura 5.1 - Processo de Testes de Sistemas Críticos
Fonte: Adaptado de (MSF, 2008).

5.2 Arquitetura da Ferramenta

A ferramenta Sistema de Apoio a Testes de Sistemas Críticos (SATS) foi criada com a estrutura vista na Figura 5.2, coerente com os modelos e processos de teste apresentados no capítulo 3 e 4. Ela fornece módulos específicos para cada uma das macroatividades do modelo. No módulo, Análise de Segurança Crítica, são identificadas e cadastradas as ameaças, as conseqüências e a probabilidade de sua ocorrência, que foram levantados e homologados pelos especialistas do domínio e usuários responsáveis. Os analistas de teste e o gerente de testes inserem no SATS os perigos, seus fatores de criticidade, FC's e os processos identificados ou criados para manter os perigos sob controle. Esse procedimento atende ao recomendado pelo MPT.BR, que é a instalação do processo Gerencia de Requisitos de Teste (GRT) (MPT.BR, 2009).

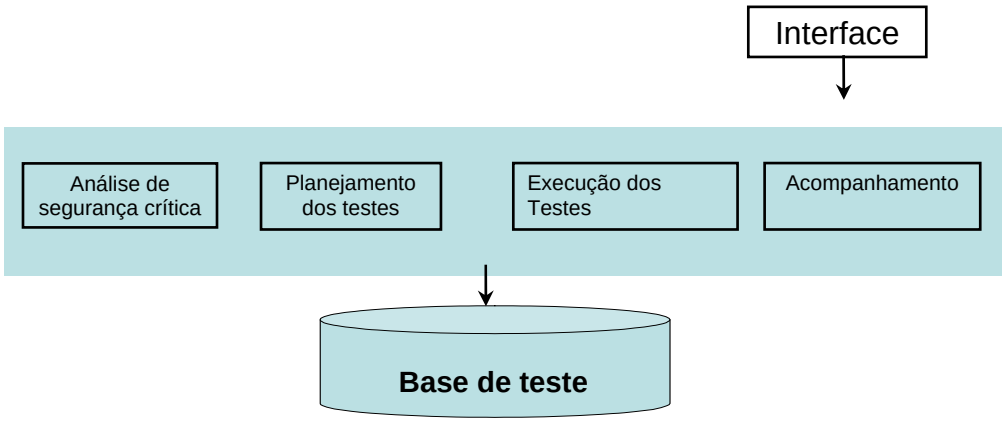


Figura 5.2 - Arquitetura da ferramenta SATS

No módulo Planejamento dos Testes Críticos, as informações obtidas no modulo anterior são acessadas para subsidiar o planejamento dos testes pela Gerencia de Projetos de Teste de Software (GPT), dentro do recomendado pelo MPT.BR e, em se tratando de sistemas críticos, as práticas específicas GPT1 a GPT12, (Quadro 4.4), que orientam a confecção do projeto de teste do sistema. O conhecimento adquirido nas atividades de análise de segurança e construção dos casos de uso e casos de testes vai sendo agregado à Base de Conhecimento, que

são algumas tabelas da própria ferramenta, possibilitando que o mesmo seja utilizado em desenvolvimentos futuros.

O terceiro módulo, Execução dos Testes, é a realização do planejado e previsto no MPT.BR no capítulo 6: Práticas Genéricas, atributo 6.1 OG1 - Executar o processo. Esse atributo é uma medida do quanto o processo atinge seu propósito. A ferramenta fornece as condições para o cadastramento dos casos de teste e orientações, para a execução dos mesmos, cadastradas em suas bases. Ela não realiza testes de software, ela apóia o planejamento e possibilita o gerenciamento de sua realização.

Tendo em vista a simplificação do processo e a necessidade de customização de cada sistema, utilizamos o banco de dados ACCESS para a prototipação das telas e para a modelagem das tabelas. O que se almeja com essa arquitetura é facilitar o treinamento de novos participantes no projeto, executar a atividade Análise de Segurança Critica com o máximo de informações disponíveis, construir o projeto de testes com maior aderência aos requisitos críticos, executar os testes do software e acompanhar o planejado e o realizado.

5.3 Requisitos

A criação da ferramenta SATS teve como objetivo fornecer condições para que os requisitos críticos, os planos de teste, casos de uso e casos de teste pudessem ser criados e disponibilizados em uma única base. A partir da literatura específica sobre planejamento e realização de testes e homologação de resultados, abordada no Capítulo 3, foi verificado que:

- Para serem confiáveis, os testes devem ser planejados e direcionados aos requisitos funcionais e não funcionais identificados. O conhecimento construído com esse levantamento e o planejamento precisam ser armazenados;
- Uma vez disponibilizado o conhecimento e sendo garantido o fácil acesso e a busca rápida da informação, os casos testes podem ser reutilizados e reexecutados;
- Busca-se um ganho de produtividade no desenvolvimento e manutenção dos softwares em geral. Ao conseguir um acréscimo na produtividade,

durante o desenvolvimento e manutenção do sistema, a qualidade e a segurança do sistema crítico também conseguem ser alavancadas;

- Uma vez armazenado, o conhecimento pode ser tratado, aprovado por especialistas e disponibilizado de forma confiável e rastreável no processo de teste (GOUVÊA, 2008);
- Uma vez disponibilizado, garantindo fácil acesso e busca rápida da informação, o conhecimento poderá ser adquirido (GOUVÊA, 2008).

Nesse contexto, foram identificados os requisitos básicos a serem atendidos pela ferramenta, a saber:

R1 - Apoiar a atividade de Análise de Segurança do Software Crítico focando nas ameaças identificadas e no seus FC's.

R2 - Gerar o processo de Teste utilizando como base outros processos de teste já existentes na ferramenta.

R3 - Permitir o planejamento de todo o processo de testes e seu acompanhamento e gerenciamento em vários níveis.

R4 - Apoiar a execução dos testes.

A ferramenta deve apoiar todas as atividades do processo proposto no capítulo anterior exploramos os requisitos R2 R3 e R4. Espera-se que a utilização da ferramenta possibilite uma maior autonomia nas várias frentes de trabalho. Na seção seguinte é exibido o Modelo de Dados da ferramenta.

5.4 DER do Modelo de Dados do SATS

A seguir apresenta-se o Diagrama de Entidades e Relacionamentos do Banco de Dados criado para operacionalizar a ferramenta. São exibidas as principais tabelas seus relacionamentos e descrição dos campos de cada uma. Para simplificar o DER foi separado em um módulo de tabelas de apoio, que é o primeiro apresentado, figura 5.3, mais 4 módulos acompanhando os principais processos do modelo.

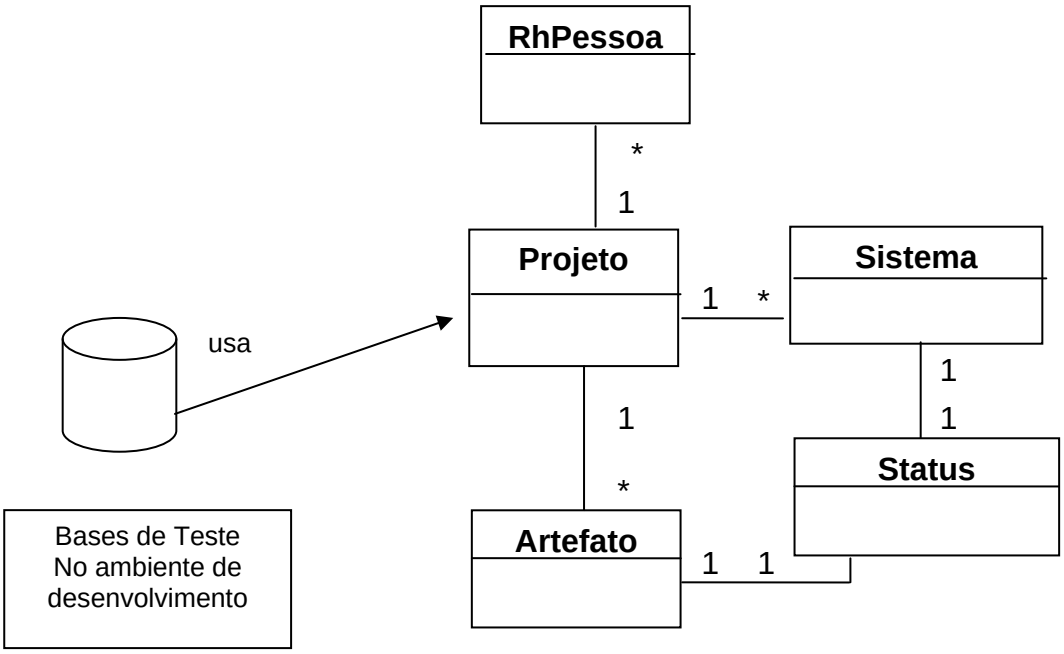


Figura 5.3 - Diagrama Parcial de Relacionamentos – Tabelas de Apoio

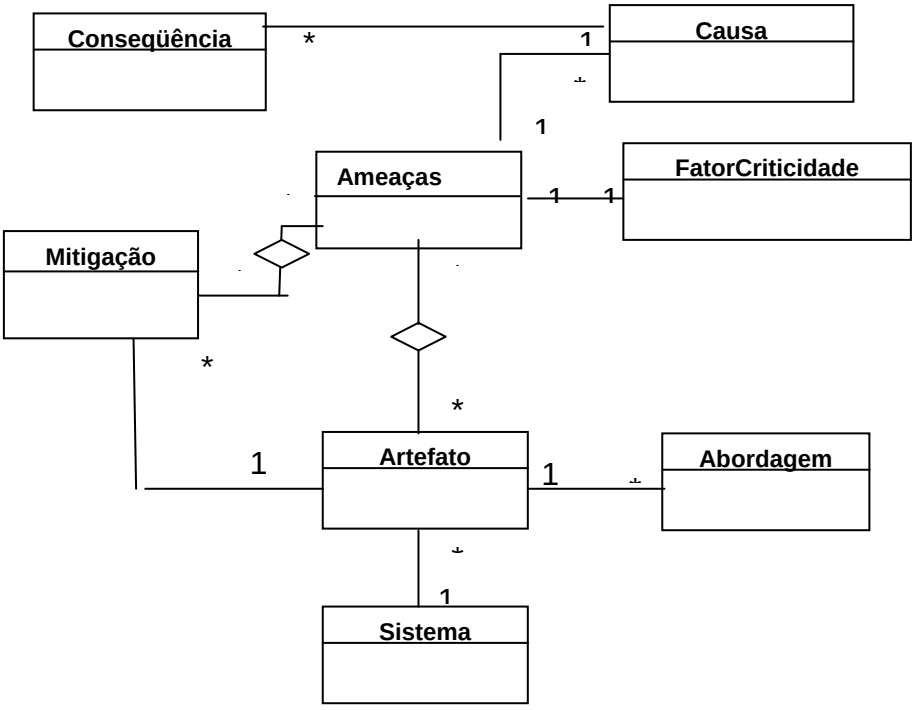


Figura 5.4 - Diagrama Parcial de Relacionamentos - Módulo de Análise de Segurança Crítica

A figura 5.4 apresenta as principais tabelas do modulo Análise de Segurança Crítica e seus relacionamentos. Nessas tabelas são gravadas as ameaças (*'Threats')* identificadas pelos especialistas e homologadas pelos usuários, assim como as causas e providências necessárias para sua mitigação.

As tabelas são preenchidas utilizando as telas do SATS. Toda a equipe pode fazer consultas às tabelas melhorando o conhecimento conjunto da equipe e permitindo uma análise crítica das definições feitas pelos especialistas do domínio e equipe de testes.

As tabelas Artefatos e Abordagem são gradualmente preenchidas e só ficam completas no módulo Planejamento de Testes, para um melhor entendimento elas foram relacionadas neste contexto. Dentro da orientação do modelo ao serem identificadas as ameaças já devem ser definidos testes para verificar e validar a sua completeza. Segue uma descrição simplificada dos principais campos das tabelas exibidas e de como as tabelas se relacionam.

Ameaça

A tabela Ameaças conterá todas as ameaças identificadas na atividade Análise de Segurança Crítica.Essa tabela deve conter a descrição da ameaça de forma sucinta assim como o fator de Criticidade ‘FC’, que a ela foi atribuído conforme quadro 2.1 e 2.2, seção 2.2 deste trabalho. A descrição deve fazer referência a exemplos e explicações mais detalhadas e técnicas necessárias para o levantamento das causas e consequências de cada ameaça identificada.

O campo status é útil porque algumas ameaças identificadas podem ser desabilitadas durante o processo de teste para que o efeito de cada ameaça possa ser isolado e testado individualmente. A facilidade de neutralizar uma ameaça mudando o seu status para inativo ou suspenso revelou-se de grande valia durante o estudo de caso.

Esta tabela se relaciona com a tabela Causa através do campo IdAmeaça. O campo é uma chave estrangeira e permite o acesso imediato de todas as causas de uma ameaça, da mesma forma com as tabelas Consequência e Mitigação. Nelas estão relacionadas as conseqüências e providencias a serem implantadas para se combater as ameaças.

Cada ameaça pode ter `n` providências de mitigação possíveis propostas. Algumas referem-se a procedimentos de segurança que devem ser implantados e auditados. Para outras podem ser processos de software a serem implementados por programas ou outros artefatos de software e/ou hardware. A identificação e o

cadastramento sinaliza a necessidade de serem tratadas as ameaças e identifica a necessidade do teste específico do procedimento eleito para a minimização.

A tabela também se relaciona com a tabela Artefatos, uma ameaça pode afetar vários artefatos que devem ser preparados para minimiza-la e controlá-la. Esse relacionamento possibilita a rastreabilidade das ameaças nos vários artefatos que executam procedimentos a elas relacionados.

Campos da tabela:
IdAmeaça - Código da Ameaça Chave Primária criada pelo Sistema (numérico)
CodSistema – Código do Sistema onde a ameaça foi identificada (texto)
CodAmeaca – Código da ameaça (texto)
DescAmeaca – Descrição resumida da ameaça (texto)
CodFtrCrt - Código do fator de criticidade classificado (texto)
Status - Status da ameaça (ativo, inativo, suspenso) (texto)

Causa

Para cada ameaça podem existir várias causas identificadas. As causas são cadastradas nessa tabela e se relacionam com as ameaças pela chave IdAmeaca. Para cada causa de uma ameaça deve-se tentar identificar pelo menos uma Conseqüência. A identificação das possíveis causas e conseqüências ajudam na definição de providências para evitá-las. Essas providências serão incluídas na tabela Mitigação.

Campos da tabela:
Id - Id da Causa, Chave Primária criada pelo Sistema (numérico)
IdAmeaça - Código da Ameaça, chave estrangeira da tabela Ameaças (numérico)
DescricaoCausa – Descrição da causa identificada pelos analistas e especialistas no domínio (texto)
Status – Status da Causa (ativo, inativo, supenso) (texto)

Conseqüência

O levantamento detalhado das possíveis conseqüências das causas identificadas permite que a consequência de maior impacto seja a escolhida para classificar a ameaça quanto ao Fator de Criticidade.

Campos da tabela:
Id – Id Consequência Conseqüência. Chave Primária criada pelo Sistema (numérico)
IdCausa - Código da Causa, Chave estrangeira (numérico)
IdAmeaça - Código da Ameaça, Chave estrangeira (numérico)
DescricaoConsequencia – Descrição da conseqüência (texto)
TipoConsequencia – 1= dano físico , 2= perda financeira, 3= perda na imagem
Categoria – 1= catastrófico, 2= Crítica, 3= moderada 4 =insignificante (quadro 2.1)
Probabilidade (A=frequente, B= provável, C= ocasional D= Remota, E= improvável)

FatorCriticidade

Esta tabela refere-se aos FC's e possibilita ter um histórico dos fatores de criticidade. Quando ocorrer alguma mudança conceitual a data de fim de validade é preenchida e o registro deixa de ser acessado e é criado outro com a data de início de validade preenchida e a data de fim aberta e esse passa a ser o registro válido.

Campos da tabela:

ID – Id do fator de criticidade, chave Primária criada pelo Sistema (numérico)
CodFtrCrt - Código do fator de criticidade (T1, T2, T3, T4)
DesFtrCrt- Descrição do fator de criticidade
DatIniValid – Data de inicio de validade
DatFimValid – data de fim de validade
Status - Status do FC (ativo, inativo, supenso)

Mitigação

Nessa tabela estão incluídas as providências definidas para se obter a mitigação das ameaças identificadas. Para providências mais complexas que afetam a vários artefatos de software ou hardware está descrito no campo DescricaoMitigaçao a referência do detalhamento da providencia.

Campos da tabela:
Id - Id da Mitigação. Chave Primária criada pelo Sistema (numérico)
IdCausa - Código da Causa, chave estrangeira (numérico)
IdAmeaça - Código da Ameaça, chave estrangeira (numérico)
DescricaoMitigaçao – Descrição da Mitigaçao (texto)

Artefato

Tabela que contêm os artefatos afetados pelas providências de mitigação, programas de software, rotinas de backup, algoritmos de criptografia de arquivos e senhas e outros definidos pelas providências de mitigação de ameaças.

Campos da tabela:
Id - Id Artefato Chave Primária criada pelo Sistema (numérico)
CodArtefato – Código do artefato (texto)
NomeArtefato – Nome do artefato (texto)

AmeaçaArtefato

Tabela que contem o relacionamento de ameaças com os artefatos que as controlam. Pode ser consultada para rastrear as ameaças a que o software crítico está exposto e os artefatos que tratam essas ameaças.

Campos da tabela:
CodAmeaca – Código da ameaça (Texto)
CodArtefato – Código do artefato (texto)
NumSeqAbor – Numero seqüencial da abordagem de Teste (numérico)

Abordagem

Tabela que contem todas as abordagens sugeridas para conter as ameaças identificadas, após análise mais detalhada ou após a implantação do sistema de software a abordagem proposta pode não funcionar ou mostrar-se ineficiente, nesse caso a data de fim de validade deve ser preenchida e o registro da tabela passa a ser um histórico que pode ser consultado sempre que necessário.

Campos da tabela:
Id - Id Abordagem Chave Primária criada pelo Sistema (numérico)
IdAmeaça - Código da Ameaça, chave estrangeira (numérico)
IdCausa - Código da causa, chave estrangeira (numérico)
DescricaoAbordagem – Descrição da Abordagem definida pela equipe de testes e especialistas no domínio
Datinival - data de inicio da validade da abordagem
Datfimval - data de fim da validade da abordagem
Status – Status da Abordagem (ativo, inativo, suspenso)

Sistema

Tabela com o código do sistema, o nome do sistema e seus status. Permite efetuar crítica de códigos válidos e exibir os nomes dos sistemas de forma padronizada e oficial da empresa.

Campos da tabela:
Id – Id Sistema Chave Primária criada pelo Sistema (numérico)
Codsistema – Código do sistema (texto)
Nomsistema- - Nome do sistema (texto)
Status – Status do sistema (Levantamento, em desenvolvimento,em testes, em homologação, em produção, inativo, suspenso)

Cronograma

Para cada Projeto, Sistema iniciado existe um cronograma a ser seguido e ele é criado nessa tabela. O acompanhamento dos prazos previstos comparados com os prazos reais possibilita a criação de métricas adequadas às equipes. Esta tabela possibilita o acompanhamento de produção da equipe e o acompanhamento gerencial da situação dos testes geral do sistema e particular do caso de teste e por artefato.

Campos da tabela:
Id - Numeração automática gerada pelo SGBD Chave (numérico)
CodCronograma, CodProcesso, CodSistema, CodTcProcedimento, DtaPrvIni, DtaPrvFim, DtaRealIni, DtaRealFim

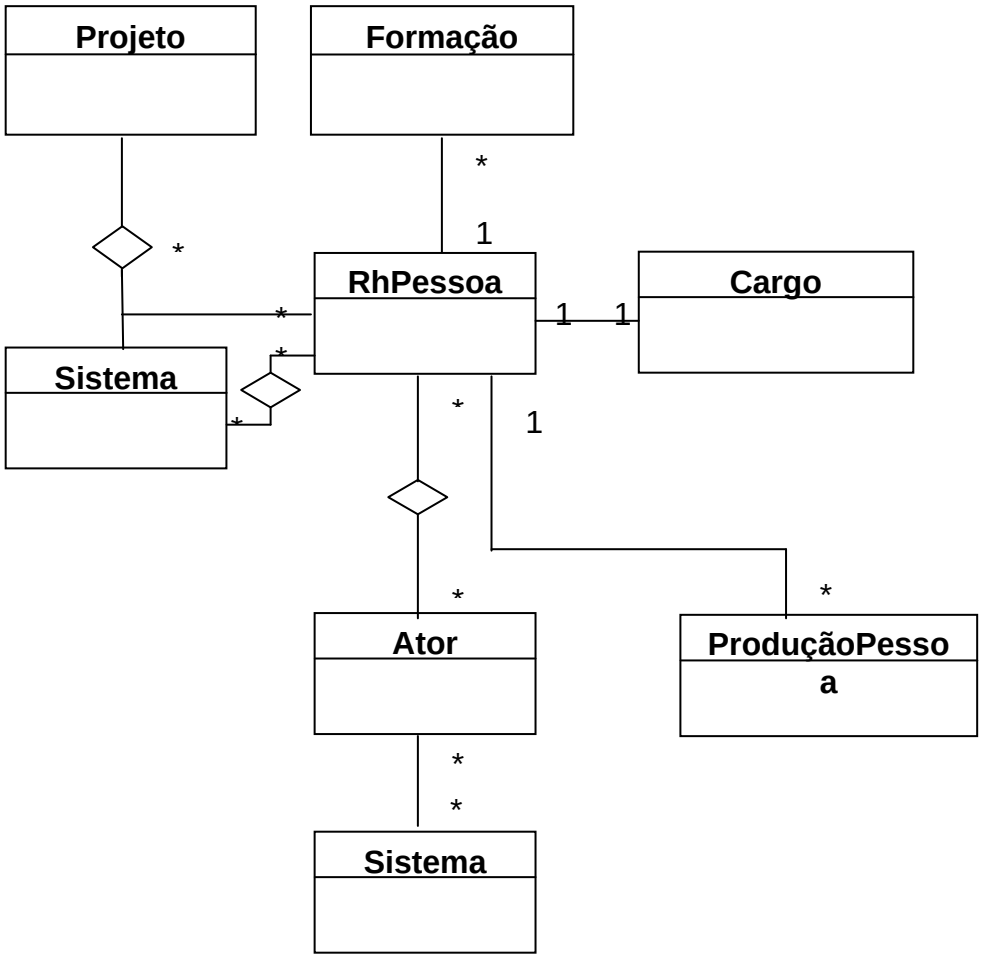


Figura 5.5 - Diagrama Parcial de Relacionamentos - Módulo Gerar o Processo de Testes (compor Equipe de Testes)

A identificação, seleção e alocação e controle de pessoas para a atividade de testes de sistemas de software críticos é uma tarefa estratégica conforme Marick, (1997). Por esse motivo ela é a primeira atividade do módulo Gerar o Processo de testes. Para apoiá-la a ferramenta SATS tem as tabelas RhPessoa onde estão cadastrados os dados que identificam todas as pessoas envolvidas com a atividade de teste do sistema, a tabela Formação permite acessar as habilidade e experiências do profissional e da equipe facilitando possíveis remanejamentos. Cada Pessoa só pode ter um cargo mas pode desempenhar vários papeis durante o processo de teste. A tabela Sistema também se relaciona com a tabela Ator e RhPessoa. Na tabela ProducaoPessoa ficam gravadas todas as atividades executadas para as quais

seja importante o acompanhamento e o controle de data prevista e data real. A seguir uma breve descrição das tabelas e dos campos de cada uma.

Pessoa

Tabela que contem todas as pessoas envolvidas no processo de testes do sistema. Analistas, testadores, usuários, homologadores, funcionários terceirizados, banco de candidatos e candidatos já pré selecionados.

Campos da tabela:

Id – Identificação sequencial da pessoa Numeração automatica criada pelo SGBD
CodPes – Código da pessoa (texto)
NomPes – Nome da pessoa
DatNasPes – data de nascimento da pessoa formato (ddmmaaaa)
CI – número da carteira de Identidade
CPFCnpj – número do cpf se pessoa física ou CNPJ se pessoa jurídica
CodCar – Código do Cargo da pessoa (texto)

Projeto

Tabela que contem dados do projeto. Os gerentes de projeto e de teste devem ajudar a definir os campos necessários para essa tabela. Durante o desenvolvimento e teste do sistema ela deve ser fonte de informações para relatórios gerenciais, apuração de resultados e comparações entre prazos previstos e realizados.

Campos da tabela:

Id - Identificação sequencial Numeração automática criada pelo SGBD
NomeProjeto – Nome do projeto texto com até 50 caracteres
CodProjeto – Código do projeto criado para identificar o projeto
DescriReduz – Descrição reduzida do projeto (texto)
DatainiPrv – data de início prevista do projeto (aaaammdd)
DatFimPrv - data de fim prevista do projeto (aaaammdd)
DatIniRealProj - data de início real do projeto (aaaammdd)
DatFimRealProj – data de fim real do projeto (aaaammdd)
StaPrj – status do projeto

PessoaFormacao

Tabela que contem os dados referente s formação de cada profissional e funcionário envolvido no projeto.

Campos da tabela:
Id – Identificação sequencial da formação numeração automática, criada pelo SGBD
CodPes – Código da pessoa (texto)
CodFor – Código da formação (texto)
DescriFor – Descrição da formação (texto)

NivelForm – Nível da Formação (1 = primário, 2 = secundário completo, 3 = superior completo, 4= Pós Graduado, 5 = Mestrado completo, 6=Doutorado Completo 7 = Especialista no Domínio)

PessoaAtor

Campos da tabela:
Tabela que contem o relacionamento da tabela pessoa com a tabela ator. Esta tabela possibilita o rastreamento dos vários papeis desempenhados por uma pessoa durante todo o processo de testes.

Id – Identificação sequencial da formação numeração automática, criada pelo SGBD
CodPes – Código da pessoa (texto)
CodAtor – Código do Ator (texto)
Datainiatr – data de início de atividade nesse papel no projeto (aaaammdd)
DatFimatr - data de fim de atividade nesse papel no projeto (aaaammdd)

Cargo

Tabela que contem os códigos dos cargos e a descrição dos mesmos. Necessária para a geração de acompanhamentos de produção e produtividade.

Campos da tabela:
Id – Identificação sequencial do cargo. numeração automática criada pelo SGBD
CodCar – Código do Cargo (texto)
Descar – Descrição do Cargo (texto)

Sistema (estrutura da tabela definida no modulo de Análise de Segurança)

Ator

No processo de análise de segurança, planejamento e criação dos teste uma pessoa pode atuar em vários papeis diferentes esta tabela permite a geração desse acompanhamento.

Campos da tabela:
Id – Identificação sequencial do Ator. Numeração automática criada pelo SGBD
CodAtor – Código do Ator (texto)
Descar – descrição do papel do ator (texto)
ValHora – valor da hora trabalhada desse ator quando atuando nesse papel

ProduçãoPessoa

Esta tabela auxilia o acompanhamento individual de atividades de cada pessoa

Campos da tabela:
CodPes - Código da Pessoa
DatHorainiatr – data de início de atividade nesse papel no projeto (aaaammdd)
DatHorfimatr - data de fim de atividade nesse papel no projeto (aaaammdd)

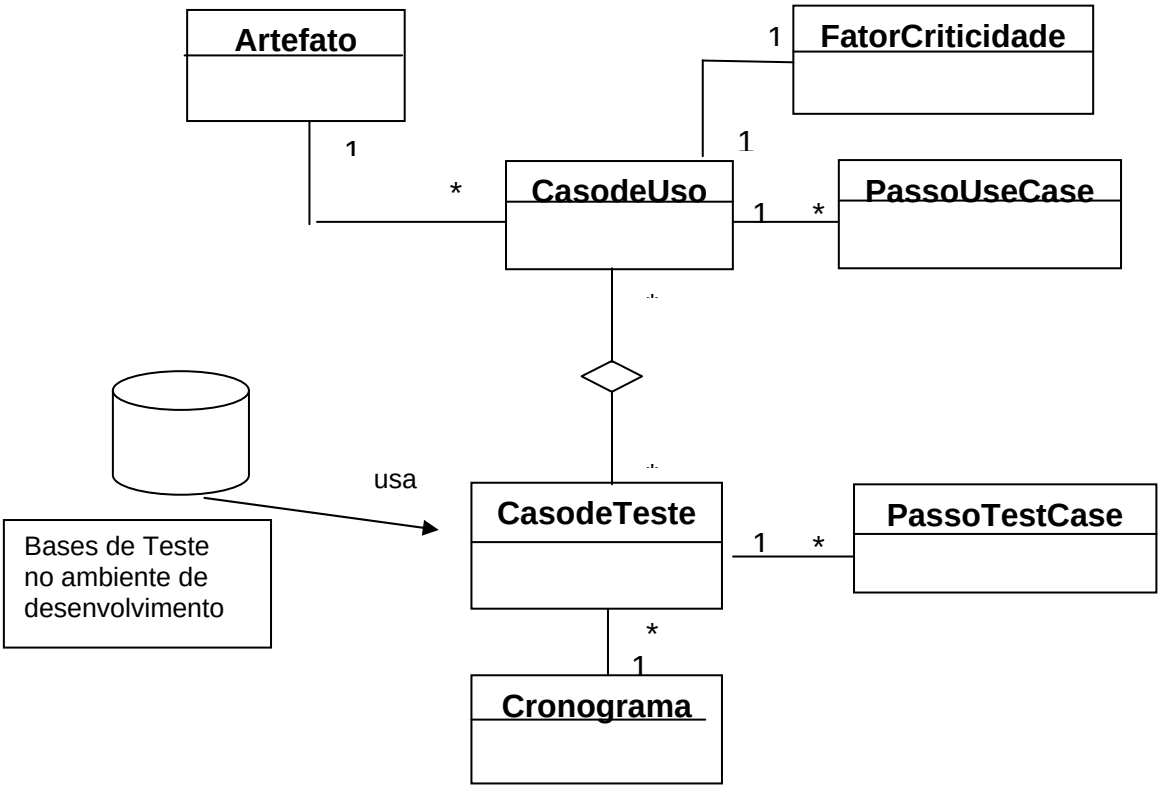


Figura 5.6 – Diagrama Parcial de Relacionamentos - Módulo Planejar Testes

Artefato

A tabela Artefato relaciona-se com a tabela caso de uso. Para cada artefato pode existir de zero a 'n' Casos de Uso cadastrados.
Campos da tabela:
Id – Numeração automática gerada pelo SGBD Chave
NomeArtefato - Texto
DescrArtefato - Texto
Status - Texto

CasodeUso

Na tabela CasodeUso estão todos os casos de uso identificados para o teste de cada artefato, nela está a explicação detalhada da função do caso de uso. Os passos do caso de uso são cadastrados na tabela PassoUseCase . A partir do caso de uso são criados os Casos de teste procurando o ponto ideal entre a cobertura do código e os requisitos críticos do sistema focando com mais rigor nos FC`s do sistema.

Campos da tabela:
Id - Numeração automática gerada pelo SGBD Chave (numérico)
CodUseCase – Código do caso de uso (Texto)

CodSistema – Código do Sistema do caso de uso (Texto)
DesUseCase – Descrição resumida do caso de uso (Texto)
TipFluxo -PCPL (principal) - ALT1 (alternativo 1) - ALT2 (alternativo 2) - ALT3 (alternativo 3) - .... ALTn (alternativo n) (Texto)
CodResCon – Código da pessoa responsável pela construção do Caso de uso
CodRev1 – Código da pessoa responsável pela primeira revisão do Caso de uso
CodRev2 - Código da pessoa responsável pela segunda revisão do Caso de uso
PreCond – Pré condição exigida para o Caso de uso se existir (Texto)
PosCond - Pré condição exigida para o Caso de uso se existir (Texto)

PassoUseCase

A tabela contem cada passo do Caso de uso, detalhando o Caso de uso e possibilitando o entendimento mais detalhado do Caso de uso necessário para o entendimento do Caso de Teste e seus passos

Campos da tabela:
Id - Numeração automática gerada pelo SGBD Chave (numérico)
CodUseCase – Código do caso de uso (Texto)
Idverifica - Identificador de verificação do caso de uso (Texto)
TipFluxo tipo do fluxo (Texto)
SeqPasso – Seqüência do passo do Caso de uso (Texto)
DscrAcao1 - Descrição do Acao ou do Resultado decorrente da ação (Texto)
DscrAcao1 – continuação da Descrição do Acao ou do Resultado decorrente da ação (Texto)

CasodeTeste

Contem para cada Caso de uso vários Casos de T este com o objetivo de encontrar o maior número de erros possível em cada artefato, nela está a explicação detalhada da função do Caso de Teste. Os passos do Caso de Teste estão cadastrados na tabela PassoTesteCase. Os Casos de teste são construídos procurando explorar os requisitos críticos e os FC`s do sistema.

Campos da tabela:
Id - Numeração automática gerada pelo SGBD Chave (numérico)
CodTestCase – Código do caso de Teste (Texto)
CodUseCase – Código do caso de uso (Texto)
CodSistema – Código do Sistema do Caso de uso (Texto)
DscTestCase – Descrição resumida do aso de Teste (Texto)
TipFluxo TipFluxo -PCPL (principal) - ALT1 (alternativo 1) - ALT2 (alternativo 2) - ALT3 (alternativo 3) - .... ALTn (alternativo n) (Texto)
CodResCon - Código da pessoa responsável pela construção do Caso de Teste
CodResRev1- Código da pessoa responsável pela primeira revisão do Caso de uso
CodRev2 - Código da pessoa responsável pela segunda revisão do Caso de uso
PreCond – Pré condição exigida para o Caso de uso se existir (Texto)
PosCond - Pré condição exigida para o Caso de uso se existir (Texto)

PassoTestCase

A tabela contem cada passo do caso de teste com as instruções, procedimentos e dados a serem informados para orientar o testador (*teste*r), na atividade de executar o teste com agilidade e quantas vezes seja necessário e também gerar as evidências de teste documentando os erros encontrados ou os resultados corretos encontrados.

Campos da tabela:
Id - Numeração automática gerada pelo SGBD Chave numérico)
CdoTestCase - Codigo do Caso de Teste (texto)
IdVerifica – Código de Verificação
TipFluxo – Tipo de fluxo PCPL (principal) - ALT1 (alternativo 1) - ALT2 (alternativo 2) - ALT3 (alternativo 3) - .... ALTn (alternativo n) (Texto)
SeqPasso – Número seqüencial de sequencia do passo
DscrAcao1 – Descrição do que deve ser feito para executar o teste (texto)
DscrAcao2 - Descrição do que deve ser feito para executar o teste (texto continuação)

Cronograma

Para cada Caso de Teste existe um cronograma a ser seguido e ele é criado nessa tabela. A cada execução dos testes são gravados dados de data e hora e situação do teste, quantidade de erros ou se correto ou não e até quantidade de erros. Esta tabela possibilita o acompanhamento de produção da equipe e o acompanhamento gerencial da situação dos testes geral do sistema e particular do caso de teste e por artefato.

Campos da tabela:
Id - Numeração automática gerada pelo SGBD Chave (numérico)
CodCronograma - Codigo do Cronograma (texto)
CodProcesso - Codigo do Processo (texto)
CodSistema - Codigo do Sistema (texto)
CodTcProcedimento - Codigo do procedimento (texto)
DtaPrvIni - data prevista de início da execução de um caso de teste (aaaammdd)
DtaPrvFim - data prevista de fim da execução de um caso de teste (aaaammdd)
DtaRealini - data real de início da execução de um caso de teste (aaaammdd)
DtaRealFim - data real de fim da execução de um caso de teste (aaaammdd)

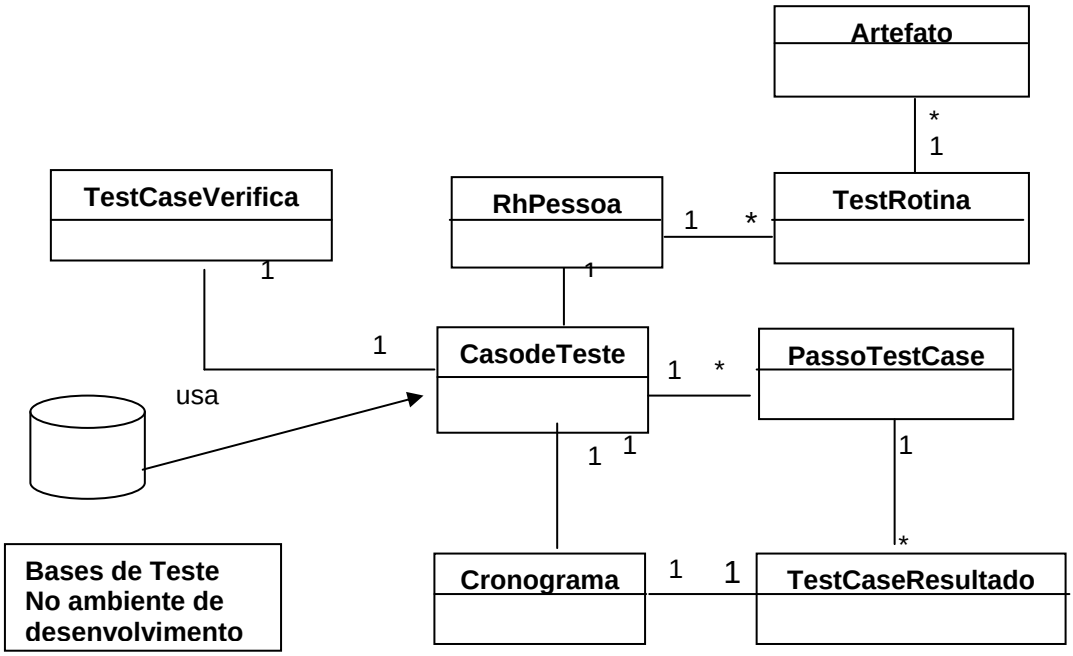


Figura 5.7 - Diagrama Parcial de Relacionamentos - Módulo Executar Testes

5.5 Utilizando a Ferramenta SATS

A seguir demonstra-se a navegação do SATS com exemplos de telas da ferramenta

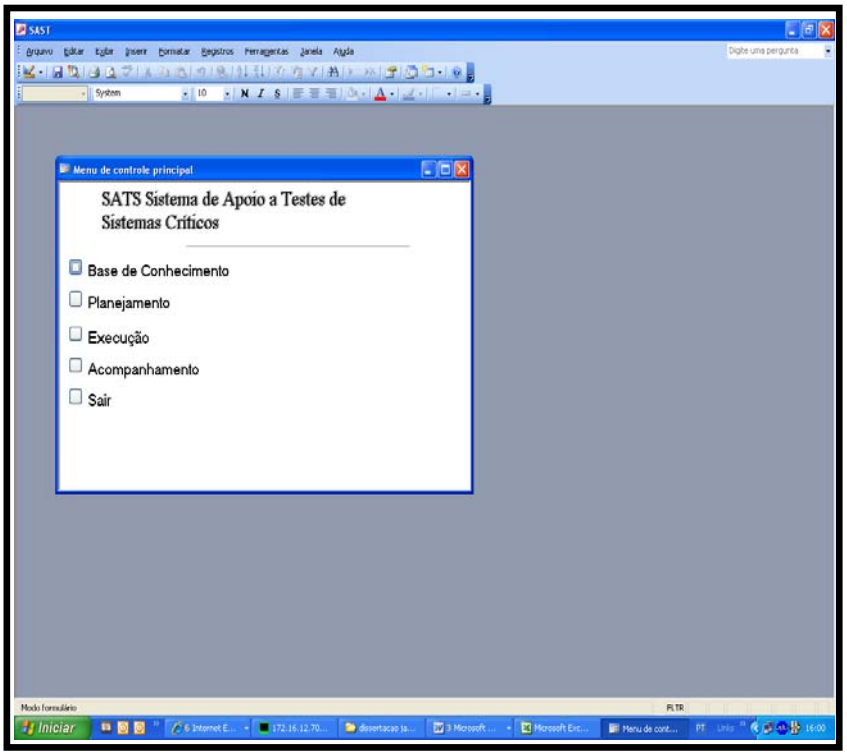


Figura 5.8 – Tela de Menu principal da SATS

A tela acima é a primeira tela de navegação do sistema. O usuário aciona o botão relativo ao módulo que deseja utilizar. Ao acessar o botão 'Base de Conhecimento' o usuário poderá consultar conteúdos diversos referentes a definições feitas nas várias fases do sistema. Poderá também incluir informações novas para serem acessadas por outros usuários. Evidências de testes corretos e errados podem ser disponibilizados neste módulo permitindo uma rápida transferência de conhecimento entre toda a equipe. Acionando o botão 'Planejamento' o sistema vai exibir telas com as opções para cadastrar o planejamento de testes ou consultar informações existentes referentes ao planejamento dos testes. O botão Execução vai navegar para um submenu, figura 5.11, que permite o cadastramento, a consulta e a execução dos casos de teste acionando os botões correspondentes. Quando o usuário for executar os testes,

conta com o auxílio das telas seguindo cada caso de teste, informando os dados necessários para a execução do teste, e comparando os resultados previstos com os resultados exibidos. O botão Acompanhamento direciona para o módulo que acompanha e gerencia o processo de testes como um todo. Várias funcionalidades podem ser oferecidas vinculando os testes aos requisitos críticos ligados às ameaças próprias do sistema que está sendo testado.

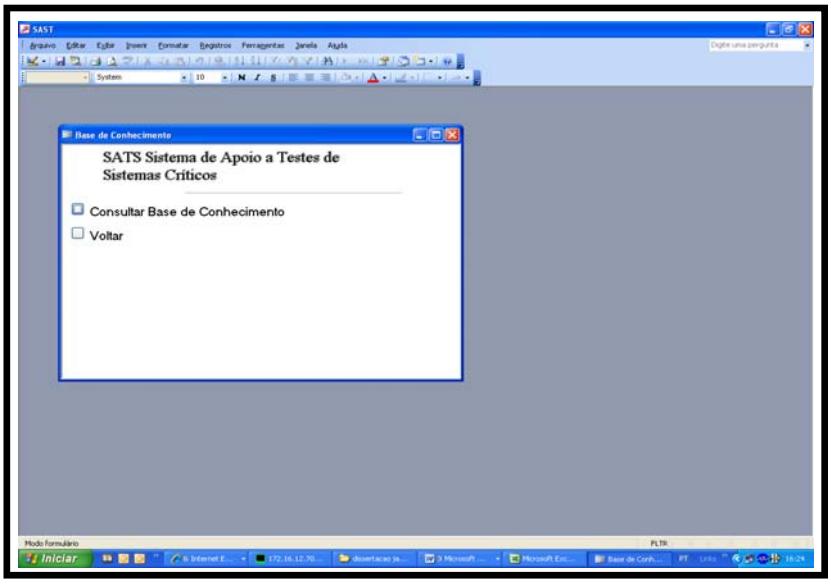


Figura 5.9 – Tela de Menu para o modulo de conhecimento.

Ao acionar o botão navega para uma lista de arquivos com conteúdos variados de conhecimento sobre o sistema

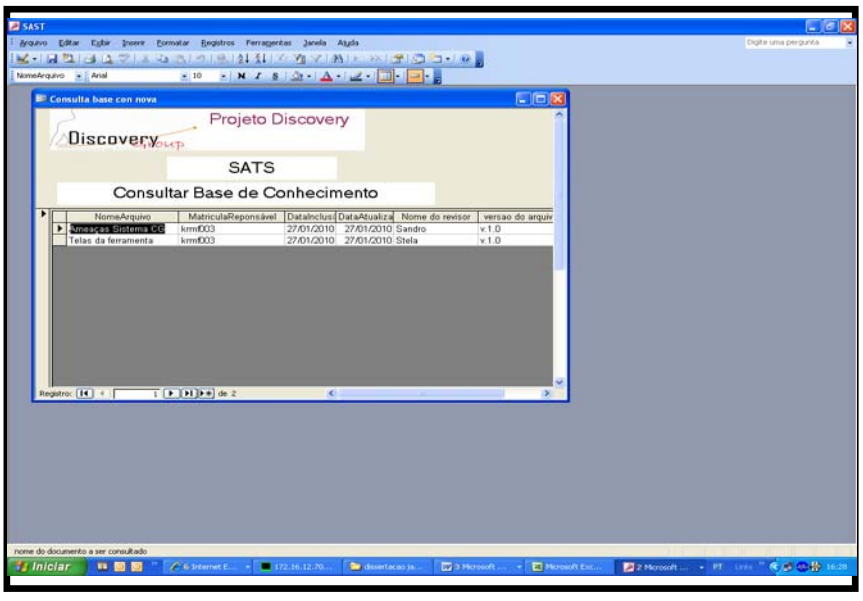


Figura 5.10 – Tela com lista de arquivos com conteúdos variados sobre o sistema

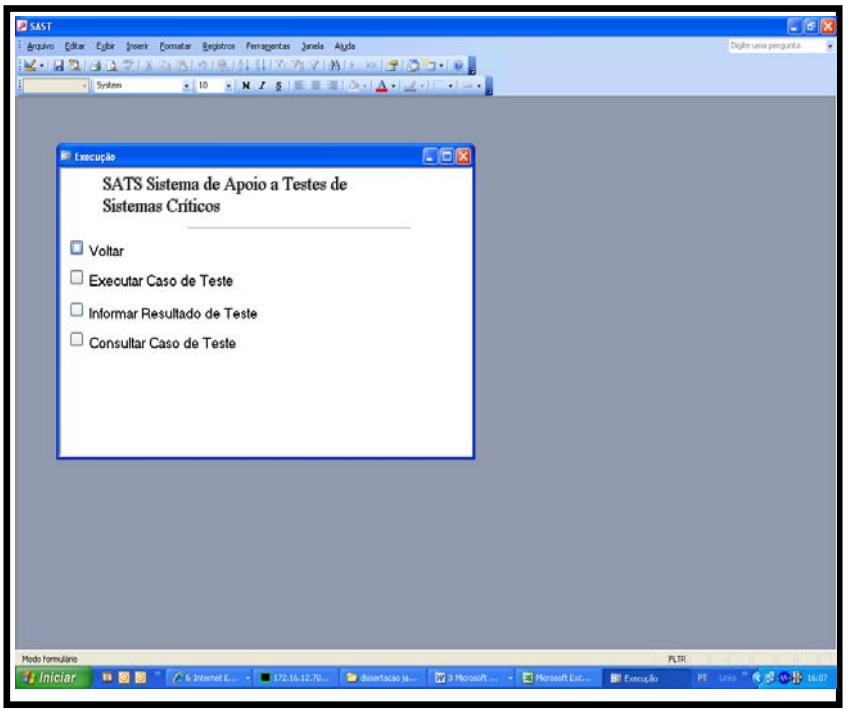


Figura 5.11. Submenu para executar casos de teste

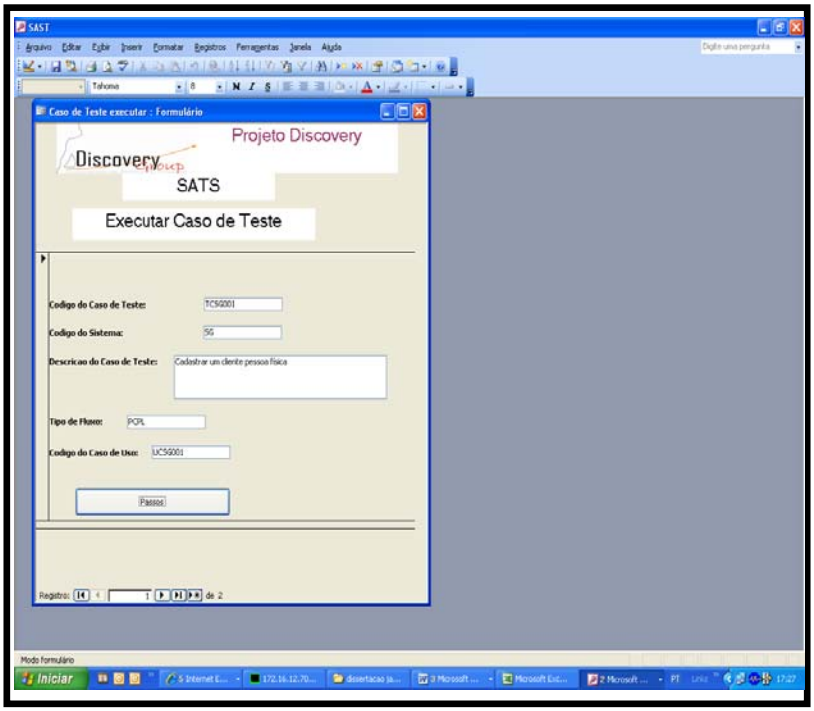


Figura 5.12. Tela Executar Caso de Teste

A figura ilustra como o Caso de Teste cadastrado permite executar caso de teste e comparar o resultado obtido com os resultados previstos.

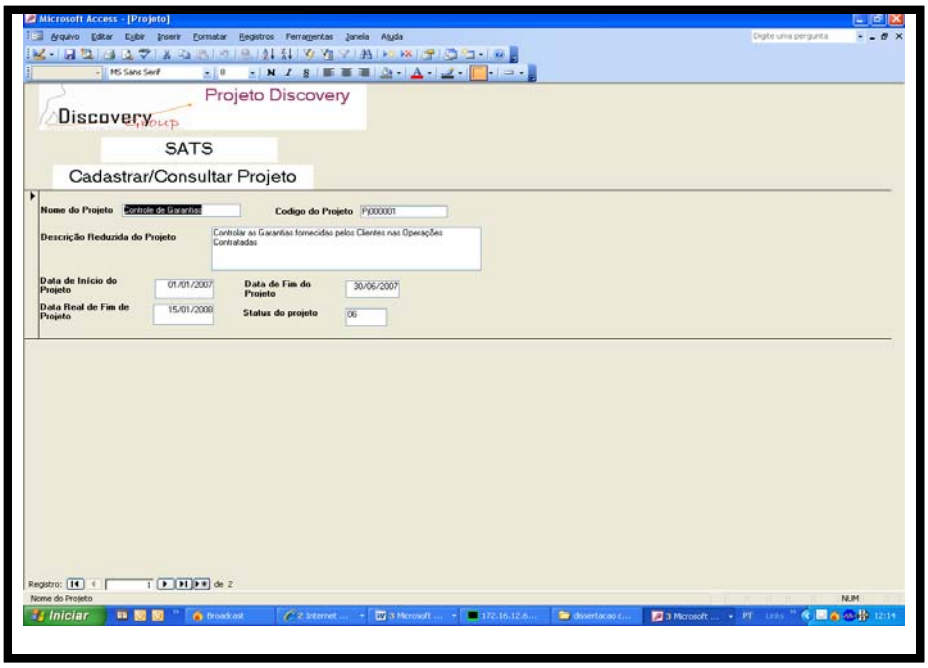


Figura 5.13 - Tela Para Cadastramento e Consulta do Projeto
(vem do submenu Planejamento)

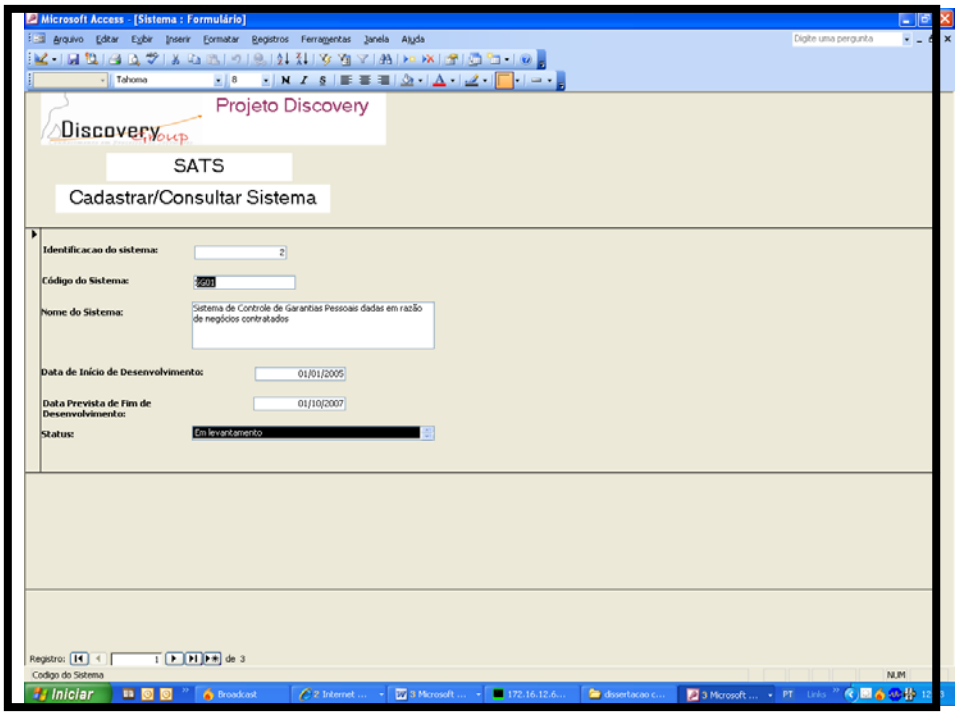


Figura 5.14 - Tela para cadastramento e consulta do Sistema Crítico
(vem do submenu Planejamento)

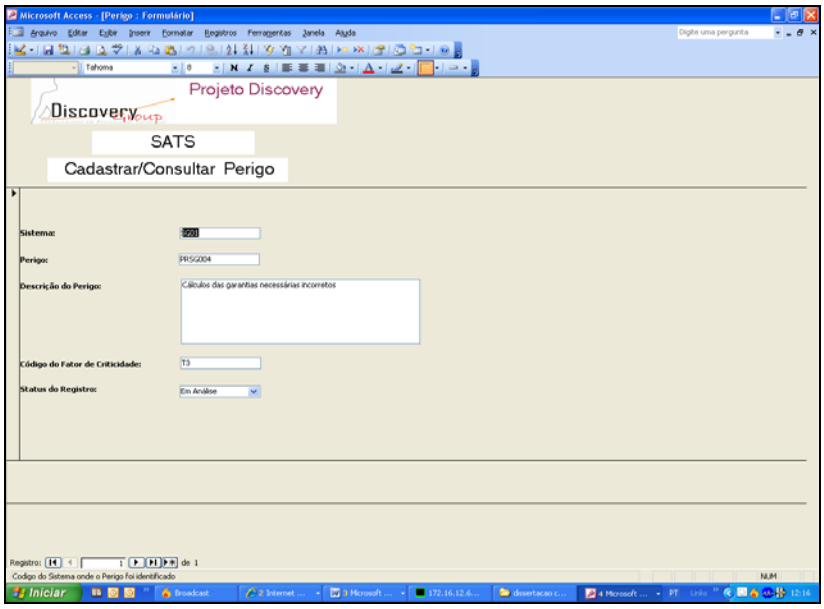


Figura 5.15 - Tela para cadastramento e consulta das ameaças (vem do submenu Planejamento)

A tela da Figura 5.15 permite o cadastramento das ameaças levantados na fase de análise de requisitos do sistema, do código do sistema, do código da ameaça, da descrição detalhada, do código do FC após a sua classificação e do status. A consulta é permitida a toda a equipe, mas a atualização é restrita aos níveis gerenciais, de análise e de homologação. Estando as ameaças cadastradas, é possível fornecer informações a nível gerencial desses perigos de forma ágil.

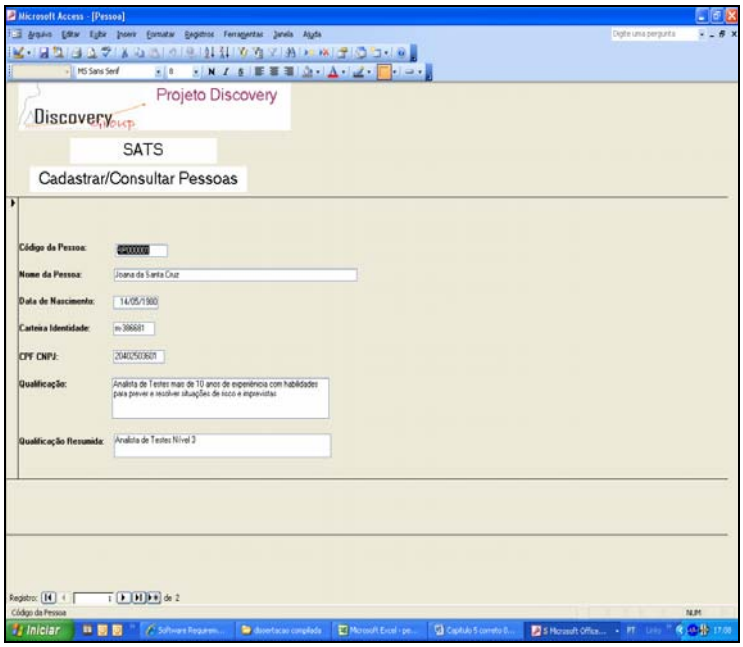


Figura 5.16 - Tela para cadastramento e consulta de pessoas (atores)

A tela da Figura 5.16 permite o cadastramento dos atores que criam os casos de uso, os casos de teste e que executam os casos de teste. Esse cadastramento possibilita o acompanhamento da produtividade dos testadores e analistas de teste. No caso de sistemas críticos, é útil poder rastrear os atores que testaram e homologaram os artefatos de missão crítica. A consulta é permitida a toda a equipe, mas a atualização é restrita aos níveis gerenciais, de análise e de homologação.

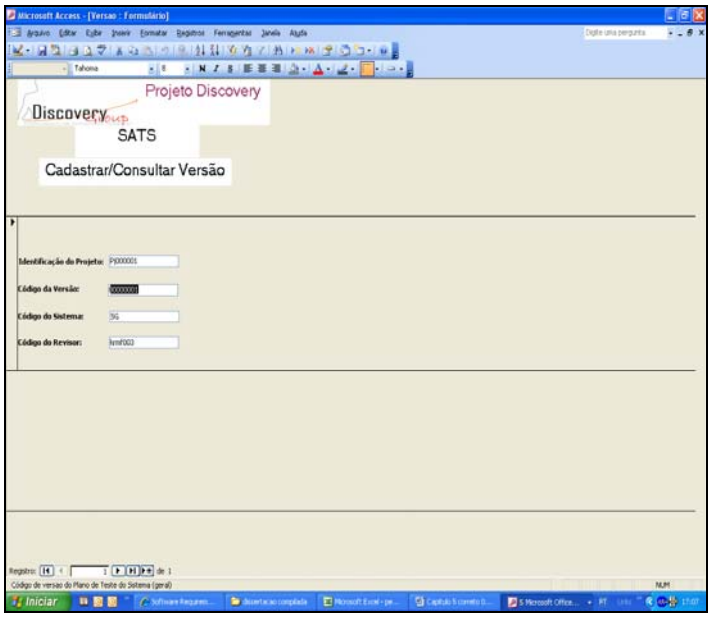


Figura 5.17 - Tela para cadastramento e consulta da versão do Sistema Crítico

A tela da Figura 5.17 permite o cadastramento da versão em que o sistema se encontra. Essa informação será útil quando o sistema for submetido a testes de regressão ou quando for necessário o processamento de versões anteriores de algum dos artefatos. A consulta é permitida a toda a equipe, mas a atualização é restrita aos níveis gerenciais e de homologação.

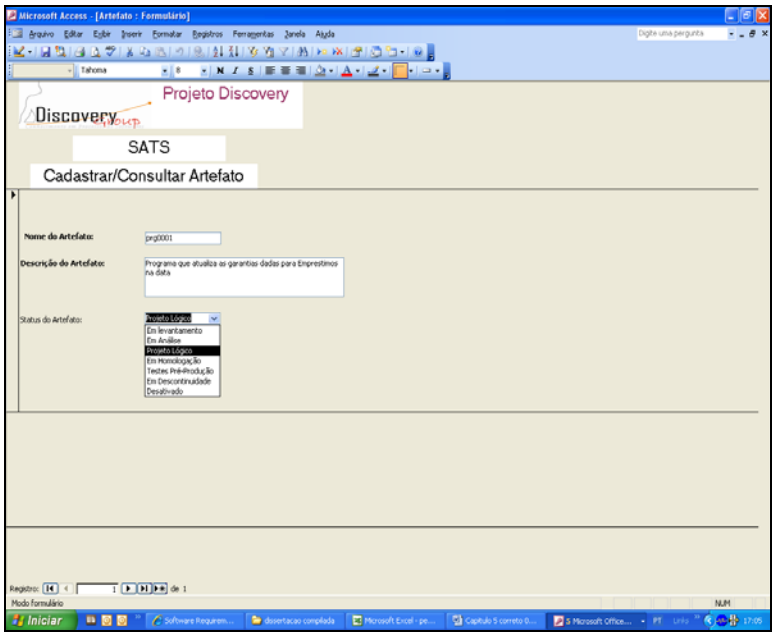


Figura 5.18 - Tela para cadastramento e consulta dos artefatos do sistema crítico

A tela da Figura 5.18 permite o cadastramento dos artefatos do sistema crítico, programas, rotinas, subrotinas e relatórios. Os analistas de teste e testadores podem consultar as funções dos artefatos quando estão executando as atividades de criação e execução dos testes. A consulta é permitida a toda a equipe, mas a atualização é restrita aos níveis gerenciais e de homologação.

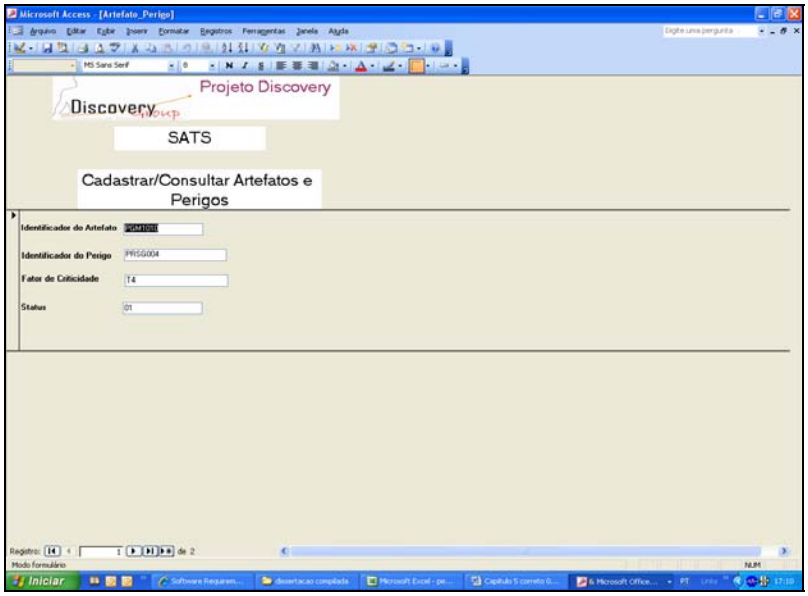


Figura 5.19 - Tela para cadastramento e consulta dos artefatos e ameaças que eles controlam

A tela da Figura 5.19 permite o cadastramento dos artefatos do sistema crítico que controlam as ameaças identificadas. Isso possibilita a rápida identificação de

artefatos que são afetados, quando da ocorrência de alguma ameaça. Permite aos analistas de sistemas, analistas de teste e testadores dimensionarem a carga de trabalho para criar e executar os testes nos artefatos que controlam as ameaças. A consulta é permitida a toda a equipe, mas a atualização é restrita aos níveis gerenciais e de homologação.

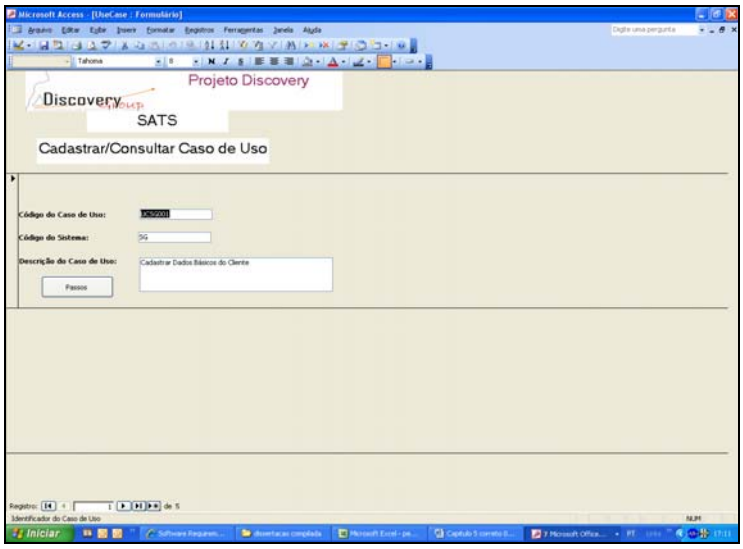


Figura 5.20 - Tela para cadastramento e consulta dos Casos de Uso dos Sistemas Críticos

A tela da Figura 5.20 será usada para o cadastramento e consulta dos Casos de Uso do sistema. Esses Casos de Uso serão a base para criação dos casos de testes para testar o sistema. O botão 'Passos' permite a navegação para os vários passos do Caso de uso. Esse cadastramento irá compor a base de conhecimento e estará disponível para a consulta por toda a equipe.

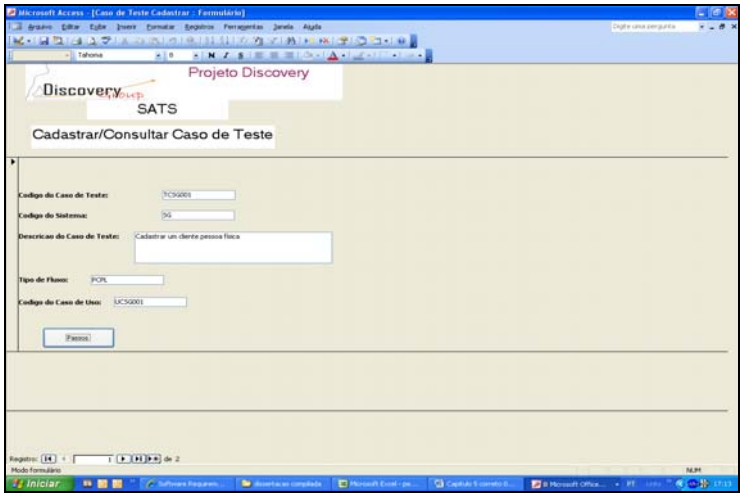


Figura 5.21 - Tela para cadastramento e consulta dos Casos de Testes dos Sistemas Críticos

A tela da Figura 5.21 exibe os Casos de Testes originados dos Casos de Uso. O botão 'Passos', permite a navegação do passo a passo do Caso de Teste, informando os dados a serem utilizados para executar o Caso de Teste e os resultados que deverão ser exibidos. Ela orienta e apóia o testador, que fica liberado da necessidade de conhecer o conteúdo das bases de teste do sistema. Aqui podem ser cadastrados Casos de Testes para os vários ambientes, Desenvolvimento, Homologação, Pré-produção e até Produção.

5.5 Conclusão

A ferramenta visa sustentar as atividades de planejamento, controle e execução dos testes de sistemas críticos, apoiando-se no conhecimento registrado na ferramenta sobre os FC’s e os artefatos que os controlam. Futuramente, a ferramenta pode ser expandida para fornecer informações sobre a qualidade dos artefatos produzidos, armazenando dados relativos à quantidade de erros por artefatos. Esse acompanhamento permitirá garantir a geração de dados apropriados para tomadas de decisão e melhoria contínua do processo de teste.

Foram analisados softwares e projetos disponíveis na literatura, dentre eles o resumo de trabalhos apresentado por Souza (2003). Entretanto, não foram identificadas, na literatura, ferramentas que associem o planejamento e construção de casos de testes, considerando os requisitos de sistemas críticos, principalmente focando nos FC's para definir a robustez dos testes planejados. Um grande diferencial de uma ferramenta seria possibilitar rastrear os artefatos que tratam os perigos identificados nesse tipo de sistema.

6 ESTUDO DE CASO DE SISTEMA CRÍTICO

Este capítulo apresenta o Estudo de Caso para o modelo proposto nesta dissertação. O Estudo de Caso foi conduzido em uma Instituição financeira de médio porte de Belo Horizonte, onde, recentemente, os testes de sistemas passaram a ser executados por uma área específica, o que possibilitou a utilização e teste do modelo proposto nesta dissertação. A identificação da empresa será resguardada bem como alguns dados, que tiveram seus valores alterados, nesta pesquisa. Isso se fez necessário para manter em sigilo informações de negócio, estratégias de mercado e produtos. A empresa será nomeada como Banco ABC.

6.1 Introdução

O sistema selecionado para fazer o Estudo de Caso foi o sistema de Controle de Garantias. O sistema cadastra e gerencia, de forma integrada e abrangente, todas as garantias oferecidas pelos clientes do Sistema Financeiro ABC em operações de crédito de forma eficiente e flexível.

6.2 Apresentação do Sistema de Garantias

O sistema funciona em duas plataformas: plataforma alta, onde estão residentes as bases de dados integradas aos outros sistemas da empresa, e plataforma baixa, que pode ser acessado pela entrante, pelos usuários/funcionários e pela internet, pelos usuários/clientes. O sistema controla vários tipos de garantia:

- Gerais - contrato e/ou nota promissória sem aval;
- Especiais - pessoais, aval, fiança, seguro ou caução; e
- Reais – hipotecária, alienação fiduciária, penhor financeiro, penhor mercantil, penhor de direito ou consignação de rendimentos, compromissos

consignação de receitas, carta compromisso, hipoteca, cheques pré-datados e duplicatas.

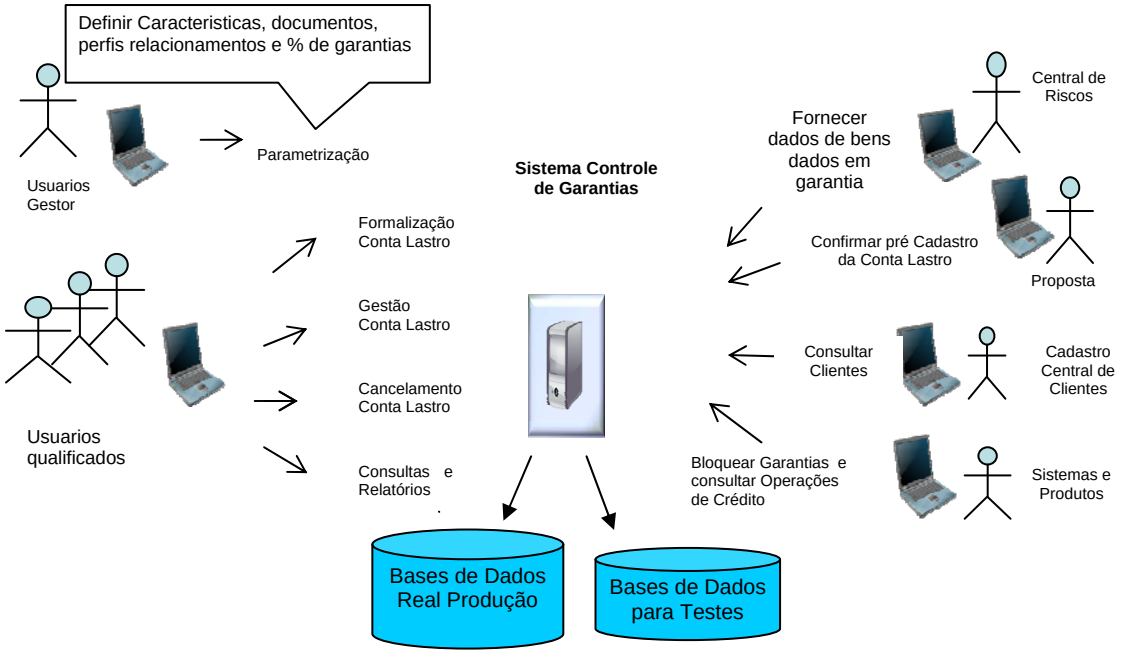


Figura 6.1 Principais módulos Operacionais do Sistema de Garantias

O sistema é composto de quatro módulos: Análise da Proposta de Crédito e Contratação da Operação; Administração das Garantias; Gestão da Inadimplência; Comunicação com o Banco Central e com o Sistema Judiciário.

As funcionalidades do sistema foram analisadas e as ameaças elicitadas e classificadas para definir os Fatores de Criticidade. Foram inventariadas 45 ameaças com fatores de criticidade T2, T3 e T4. No quadro 6.1 relaciona-se 9 ameaças como exemplo. A análise das ameaças e de suas possíveis consequências resultou na classificação do sistema como crítico. Outros conceitos do sistema e seus processos estão detalhados de forma resumida no Anexo 6.

6.3 Ameaças identificadas no Estudo de Caso

Sequ	Ameaça	Possíveis Causas	Tipo	FC	Probabilidade	Class. Categoria
1	Avaliação do potencial de endividamento a maior	bens informados a maior	S	T2	remota	II
2	Avaliação do potencial de endividamento a menor	bens informados a menor	S	T2	remota	II
3	Liberação do recurso financeiro com mais de 48 horas	Demora de análise pelo Comitê de Crédito	H	T3	Provável	II
		Cadastro desatualizado	H	T3	Provável	II
4	Sistema de Controle de Garantias indisponível por mais de 15 minutos	Rede sobrecarregada	S	T3	Provável	III
		Manutenção no sistema	H	T3	Freqüente	III
5	Avaliação das aplicações financeiras do cliente incorretas	Sistema que atualiza valores incorreto	S	T4	Freqüente	II
		Novas aplicações não foram informadas	H	T3	Provável	II
		Resgates de aplicações bloqueadas efetivados indevidamente	S	T3	Freqüente	III
6	Fornecimento de informações incorretas para os sistemas integrados	Sistema de controle de garantias com erro	S	T3	Ocasional	II
7	Fornecimento de informações do cliente incorretas para o Banco Central	Manutenção nos dados do cliente não capturada pelo sistema de CG.	S	T3	Ocasional	I
8	Não efetivar bloqueio judicial solicitado pelo Tribunal de Justiça	Arquivo eventual enviado pelo TJ não processado	H/S	T3	Ocasional	I
09	Cálculos de valores errados	Fórmulas definidas incorretamente ou não atualizadas devido a mudanças legais	H	T4	Provável	I

S=Falha de Sistema S=Falha Humana

Quadro 6.1 Ameaças identificadas no sistema de Controle de Garantias

6.4 Utilizando a Ferramenta no Sistema Crítico

Foi possível conseguir a aprovação da empresa para utilizar o modelo proposto para executar os testes do sistema, porque o sistema estava sendo desenvolvido pela empresa e é considerado estratégico para a instituição. A empresa já tinha passado por várias experiências de atrasos na implantação de sistemas fornecidos por consultorias e desenvolvidos internamente com histórico de atrasos na etapa de testes. Existia um histórico de sucessivos erros em sistemas implantados com falhas e posteriormente diagnosticados como indevidamente testados.

As principais características do sistema são: a necessidade de funcionar 24 horas por dia, 7 dias por semana acessível aos usuários e clientes pela rede mundial de computadores.

Seguindo o modelo proposto, foi feita, inicialmente, a atividade de Análise de Segurança do Software a ser desenvolvido. Foram levantadas as ameaças a que o sistema estava exposto, suas possíveis causas conforme ilustrado no quadro 6.1. Foi também feita uma classificação quanto ao tipo da causa usou-se o indicador 'S' para as causas originadas pelo sistema e 'H' para causas humanas. As ameaças identificadas foram relacionadas no formulário Inventário de ameaças e foram disponibilizados na base de conhecimento da empresa. Foi utilizado, nas análises iniciais, o modelo de levantamento sugerido no Anexo 2. Para os requisitos de qualidade do sistema, foram definidos os requisitos funcionalidade, usabilidade, confiabilidade, manutenibilidade eficiência e portabilidade (ISO/IEC, 9126) a serem considerados nos testes a serem criados.

Cada ameaça, após ser entendida e analisada pela equipe, foi classificada quanto ao Fator Crítico 'FC'. Essa classificação foi feita utilizando a faixa de probabilidade de ocorrência e a severidade de sua conseqüência, conforme os Quadros 2.1 e 2.2, definidos no capitulo 2. As ameaças identificadas após serem homologadas pelos usuários e especialistas no domínio foram cadastradas no SATS. Para o estudo de caso desse trabalho, foram selecionadas as ameaças 5, 7 e 9 listadas no Quadro 6.1. As principais telas da ferramenta utilizadas nesse estudo de caso são exibidas e seu funcionamento e utilização explicadas brevemente a seguir:

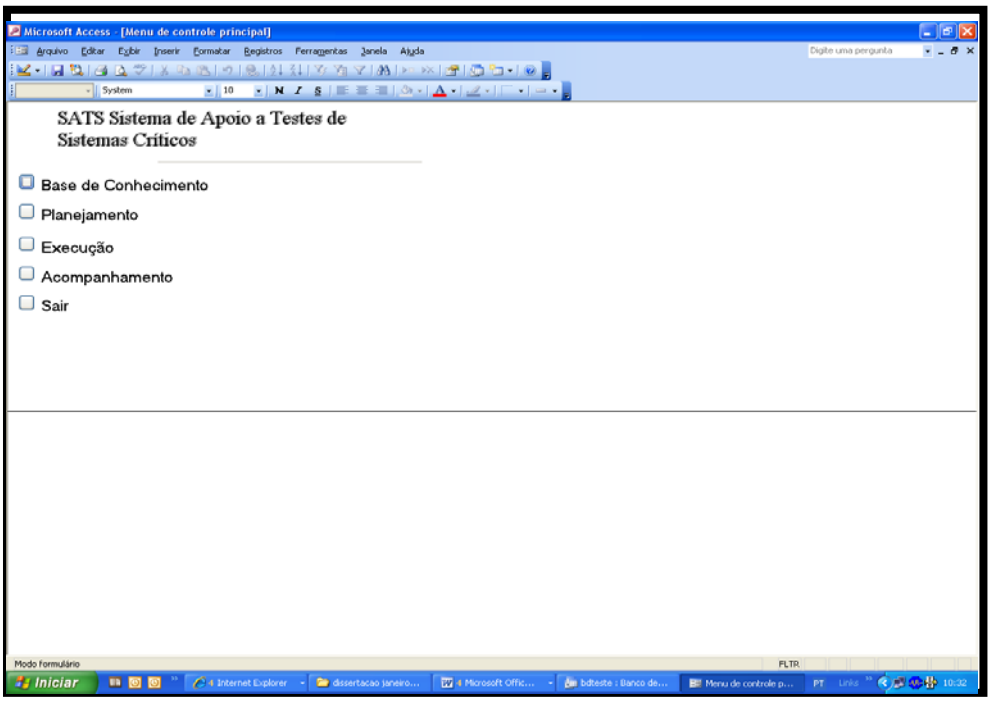


Figura 6.2 – Tela Menu Principal

A figura 6.2 acima ilustra como iniciar o uso da ferramenta. Ao acionar na tela o botão base de conhecimento é possível consultar informações sobre outros sistemas testados, exemplos de Casos de Uso, Casos de Testes e outros conhecimentos. O botão Planejamento ao ser acionado direciona para um menu que navega para a Especificação do Projeto de Testes e o Cadastramento dos Casos de Teste entrando na macroatividade Gerar o Processo de Teste. Porém antes disso é necessário preencher as tabelas auxiliares. Isso pode ser feito acessando diretamente as telas específicas para cada umas delas. A título de exemplo são exibidas a seguir: a tela para cadastramento do sistema e a tela para cadastramento de pessoas.

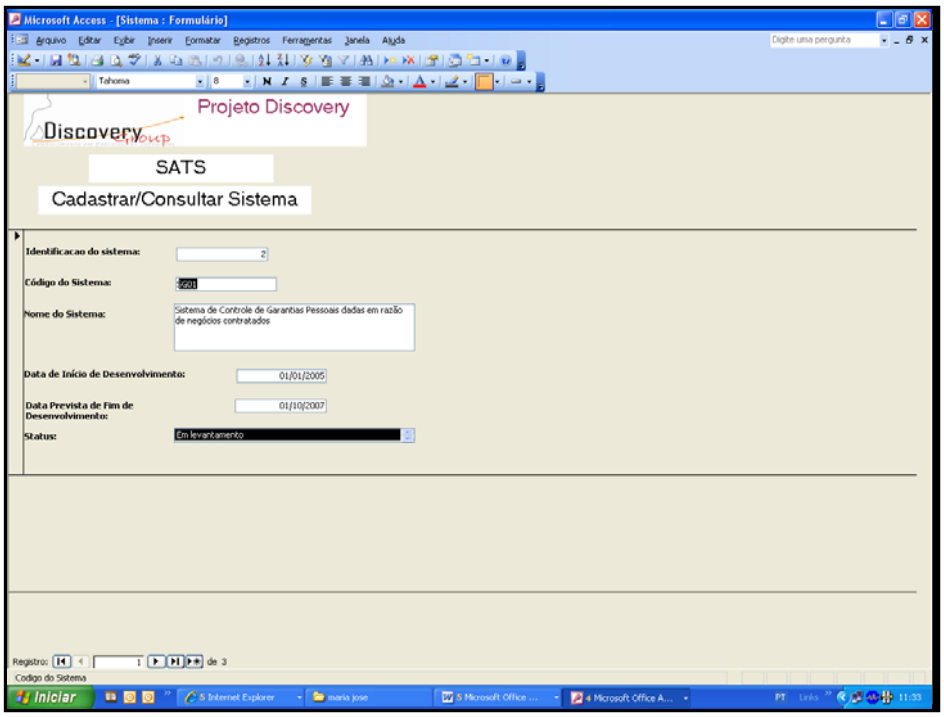


Figura 6.3 – Tela para cadastramento do sistema a ser testado

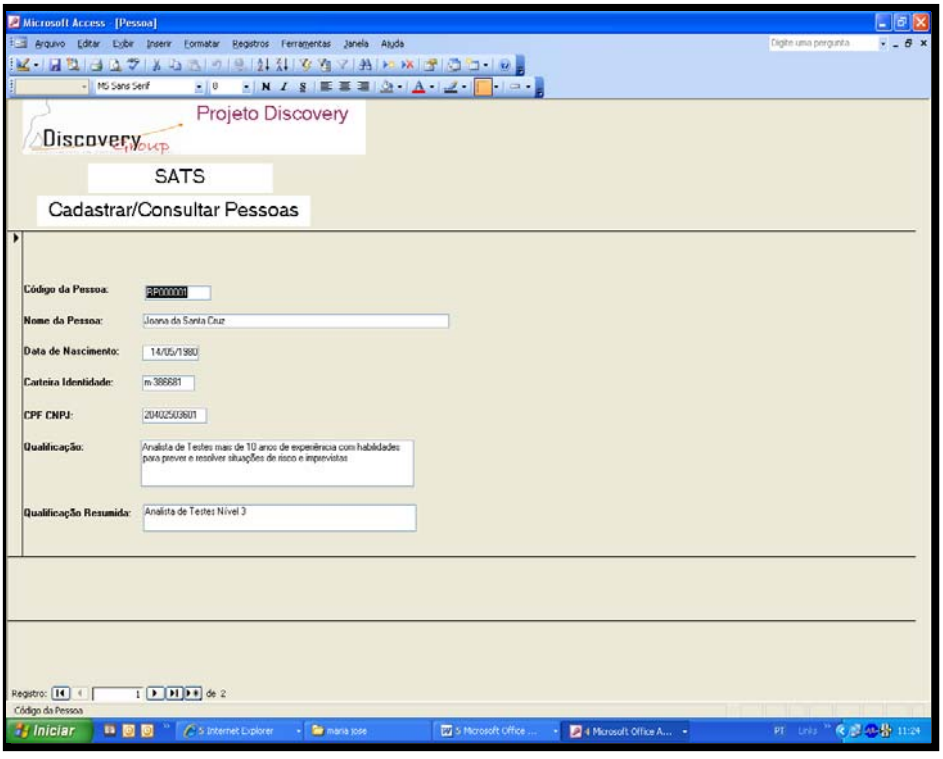


Figura 6.4 – Tela para cadastramento de pessoas integrantes da equipe

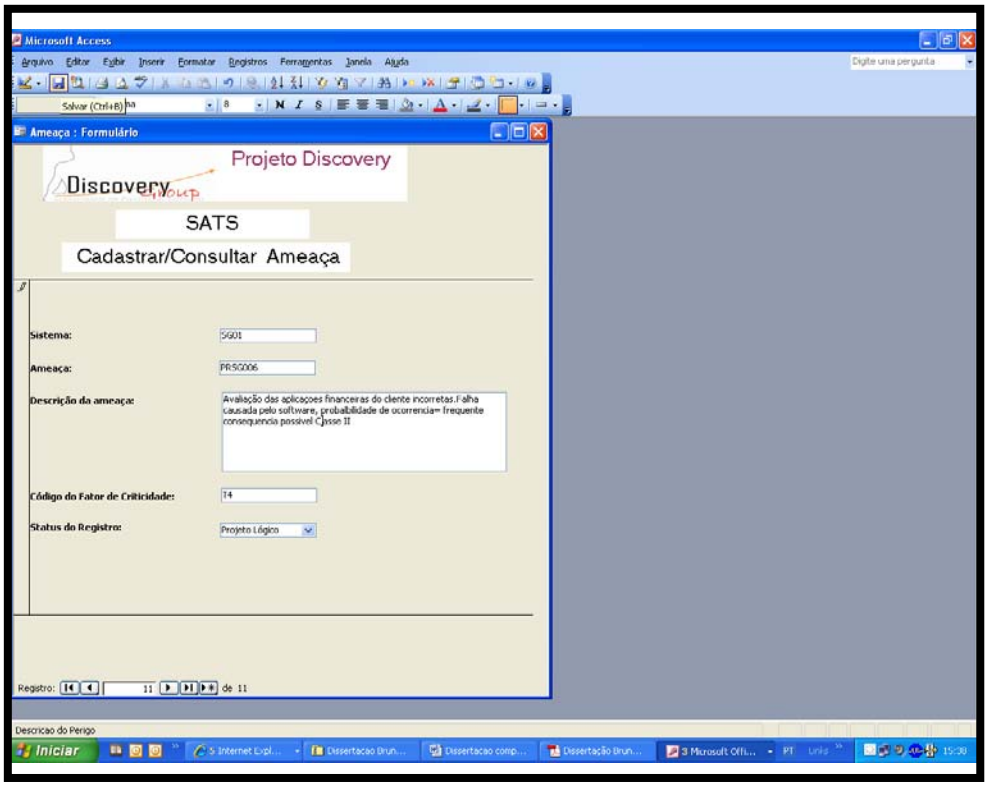


Figura 6.5 – Cadastramento de ameaça elicitada

A figura 6.5 ilustra o cadastramento de uma ameaça escolhida para exemplificar o caso de uso detalhado como exemplo, nessa tela também é informado o FC da ameaça e se ela é deflagrada por um software ou pelo uso humano, a probabilidade de sua ocorrência e a gravidade de sua possível conseqüência. As vantagens de cadastrar as ameaças são a facilidade de acesso por toda a equipe e agilidade em manter as informações atualizadas.

Para acompanhamento do modelo, escolhemos detalhar a ameaça PRSG006, Avaliação das aplicações financeiras do cliente incorretas, listada no quadro 6.1, sequência = 5. A ameaça tem o FC = T4 o e a construção do caso de uso e caso de teste relacionado a essa ameaça é mais simples. Durante a experimentação do modelo, foram construídos casos de uso detalhados utilizando um framework exemplificado no anexo 5 para cada ameaça identificada.

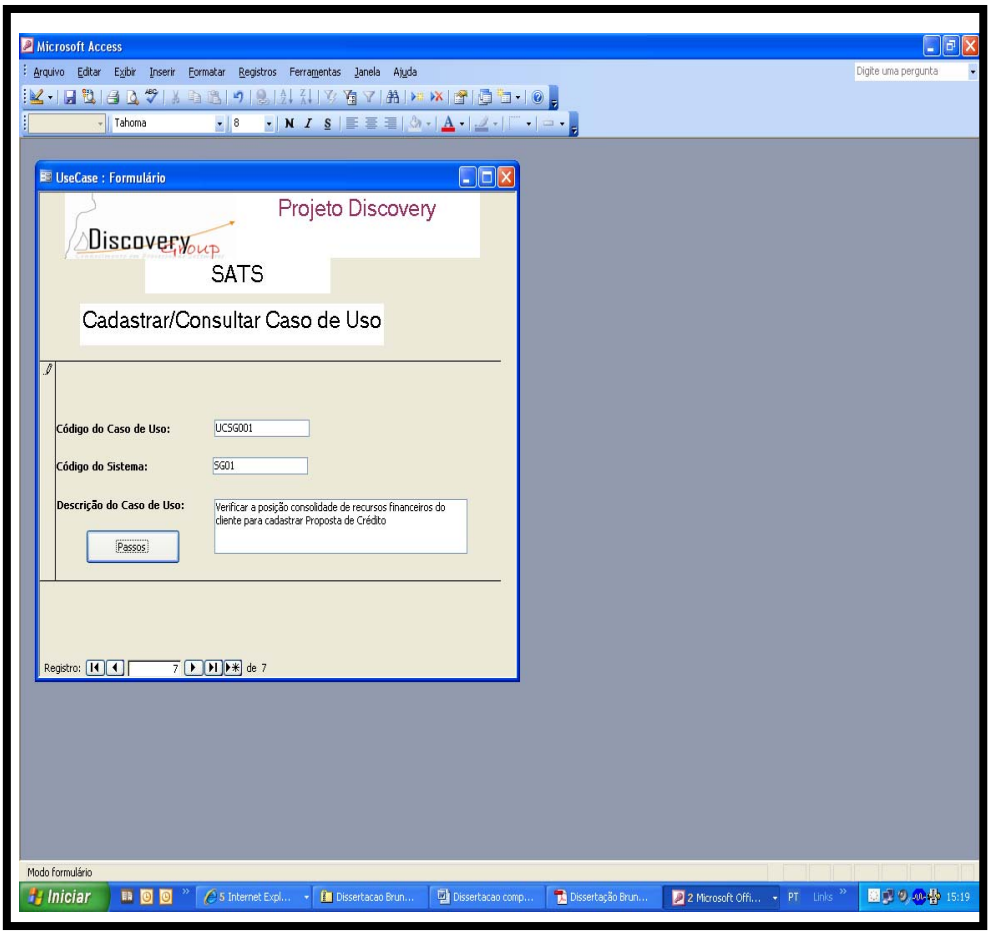


Figura 6.6 – Cadastramento de Caso de uso relacionado a ameaça elicitada

A figura 6.6 ilustra o cadastramento do caso de uso relacionado a ameaça utilizada como exemplo. A tela do SATS possibilita o cadastramento do código do caso de uso, o código do sistema, uma descrição breve do caso de uso e exibe um botão 'Passos' que ao ser acionado navega para a tela seguinte para que sejam cadastrados quantos passos tenham sido identificados no caso de uso.

Como demonstra-se a seguir:

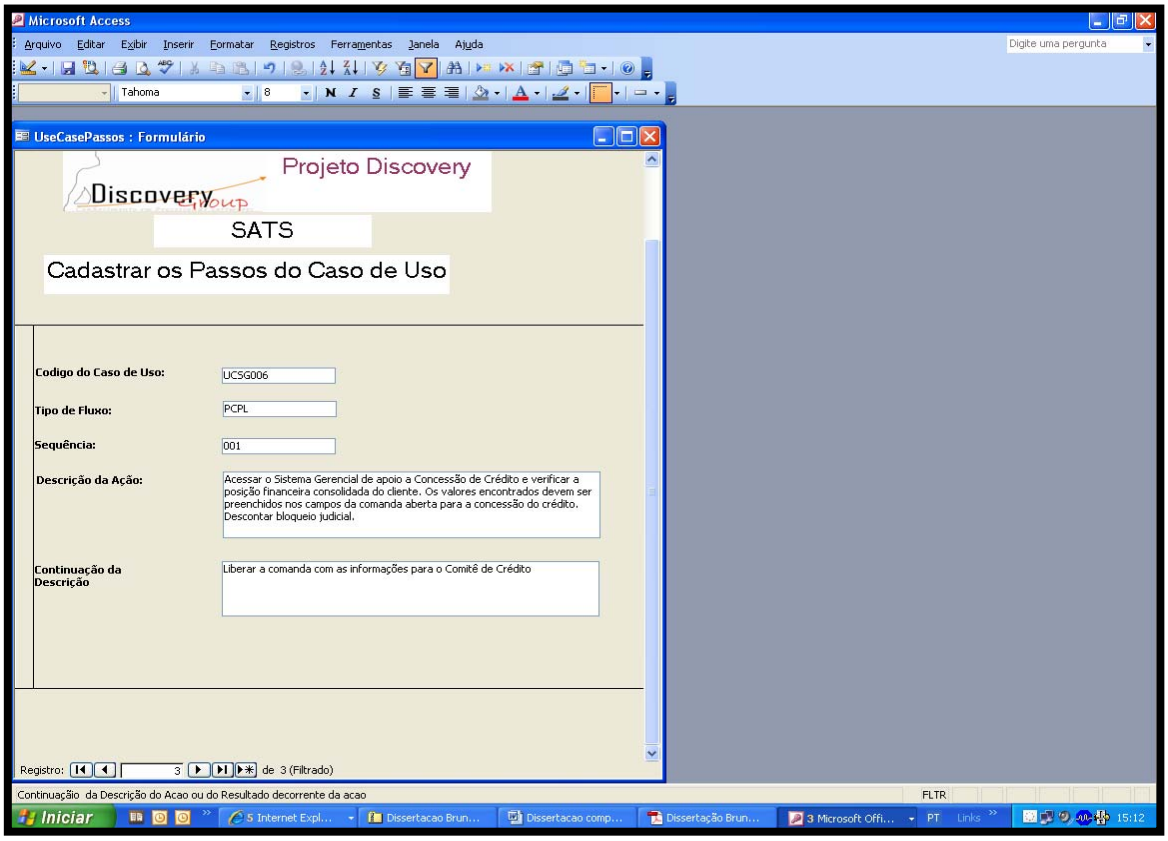


Figura 6.7 – Cadastramento dos Passos de caso de uso relacionado à ameaça

A figura 6.7 ilustra o cadastramento dos passos do caso de uso. A possibilidade de cadastrar vários passos para um caso de uso permite contemplar os vários caminhos e fluxos alternativos do caso de uso. Isso facilita o entendimento do processo pelos integrantes da equipe. É muito importante entender o caso de uso porque dele derivam os casos de testes que serão construídos na macroatividade Planejar Testes e processados na macroatividade Executar Testes. Por simplificação só o cadastramento de um passo será exibido.

Uma parte significativa do sistema já tinha sido desenvolvida, mas a identificação dos Fatores de Criticidade teve como conseqüência algumas alterações nos programas já prontos com o objetivo de adequá-los à mitigação das ameaças identificadas. Enquanto os artefatos estavam sendo ajustados, as equipes de construção e teste do sistema seguiram em frente para desenvolver as atividades seguintes propostas pelo modelo.

Na segunda, Gerar o Processo de Testes foi definida a estratégia a ser utilizada para testar o sistema. Todos os requisitos críticos foram cadastrados na ferramenta e os programas e rotinas relacionados a cada requisito crítico identificados. Esse levantamento foi revisado pelos usuários, equipes de testes e

desenvolvimento, obtendo de todos os envolvidos a homologação dos mesmos. Os analistas de testes acompanharam os analistas de sistemas nos levantamentos feitos na fase de identificação de requisitos e cadastraram na ferramenta todos os casos de uso gerados.

Os analistas construíram os Casos de Uso relacionados com as ameaças. Os Casos de uso foram verificados pelos analistas de testes, que analisaram a testabilidade de cada um. Os casos de teste foram construídos com base nos casos de uso homologados. As figuras 6.7 e 6.8 a seguir ilustram o cadastramento de artefatos e do caso de teste gerado a partir do caso de uso ilustrado acima.

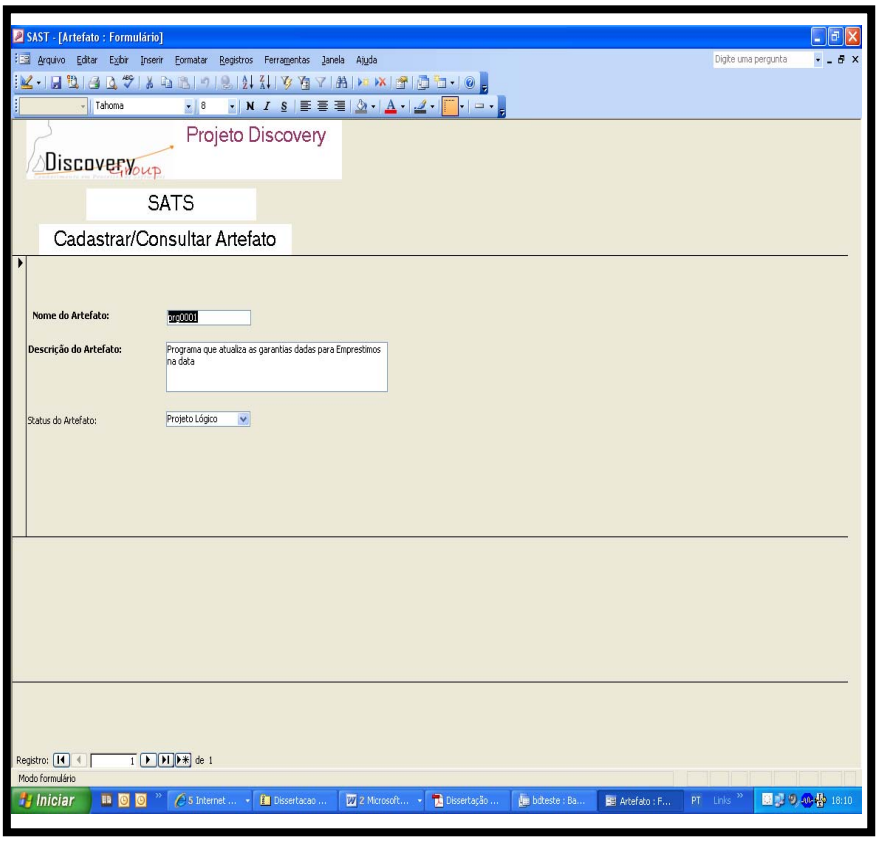


Figura 6.8 – Cadastramento de artefatos relacionados à ameaça a serem testados com rigor

O artefato prg001 é um programa de software, que atualiza diariamente as garantias dadas em operações de crédito aos clientes. Ele é o responsável pelo cálculo dos valores disponíveis dos clientes para garantir os recursos financeiros solicitados. Ele foi identificado como um artefato que deve ser rigorosamente testado porque os valores de bens e recursos financeiros que ele pesquisa nos sistemas corporativos influem diretamente na liberação dos recursos para o cliente. A sua

função diária é pesquisar nas bases corporativas e identificar os recursos do cliente que encontram-se pulverizados em várias aplicações identificando possíveis bloqueios judiciais, resgates de aplicações pela WEB ou identificação do cliente como pessoa politicamente exposta. Por esse motivo exibiremos o Caso de Teste criado para testar de forma robusta o artefato de modo a mitigar a ameaça identificada.

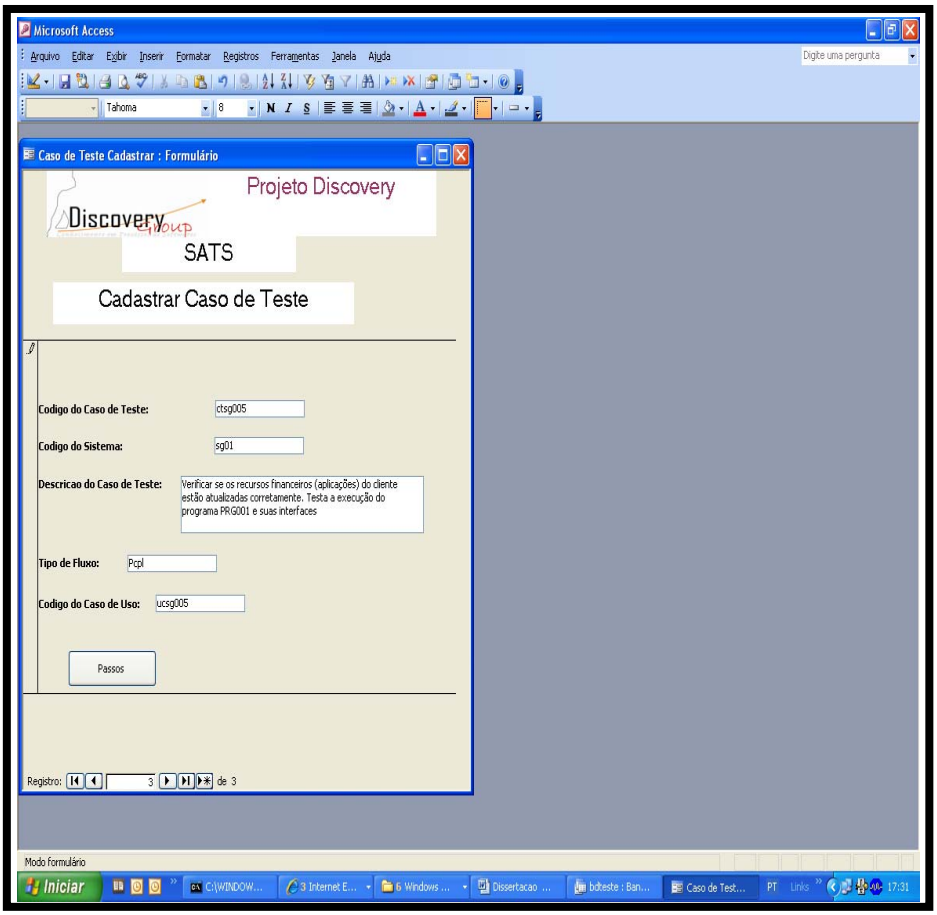


Figura 6.9 – Tela para Cadastramento do caso de teste criado para testar a ameaça

A tela possibilita o cadastramento de uma explicação da função do caso de teste, o código do caso de teste, o código do sistema, o tipo do fluxo e o código do caso de uso que originou o caso de teste. Os passos a serem seguidos para executar o caso de teste serão cadastrados na tela seguinte ao ser acionado o botão 'Passos'.

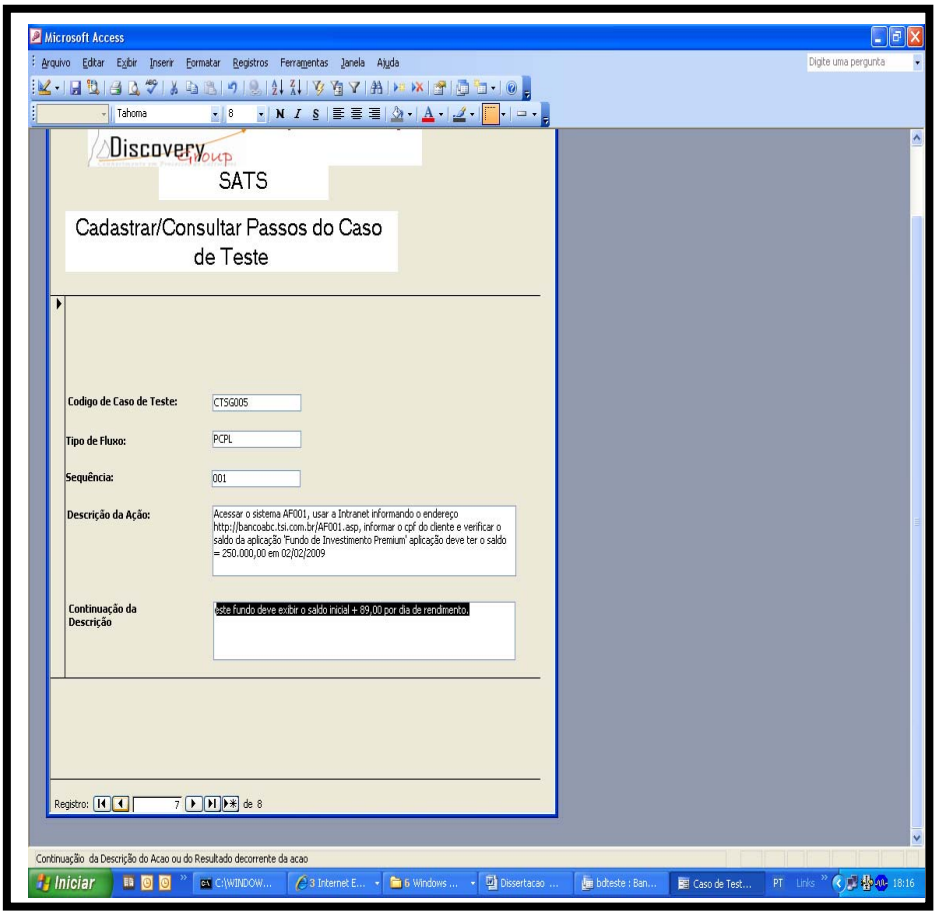


Figura 6.10 – Tela para Cadastramento do passo 1 do caso de teste

O cadastramento de cada passo do caso de teste deve informar todos os caminhos e dados a serem informados ao sistema que vai ser testado assim como os resultados esperados. Devem ser feitos casos de testes considerando resultados corretos e resultados incorretos.

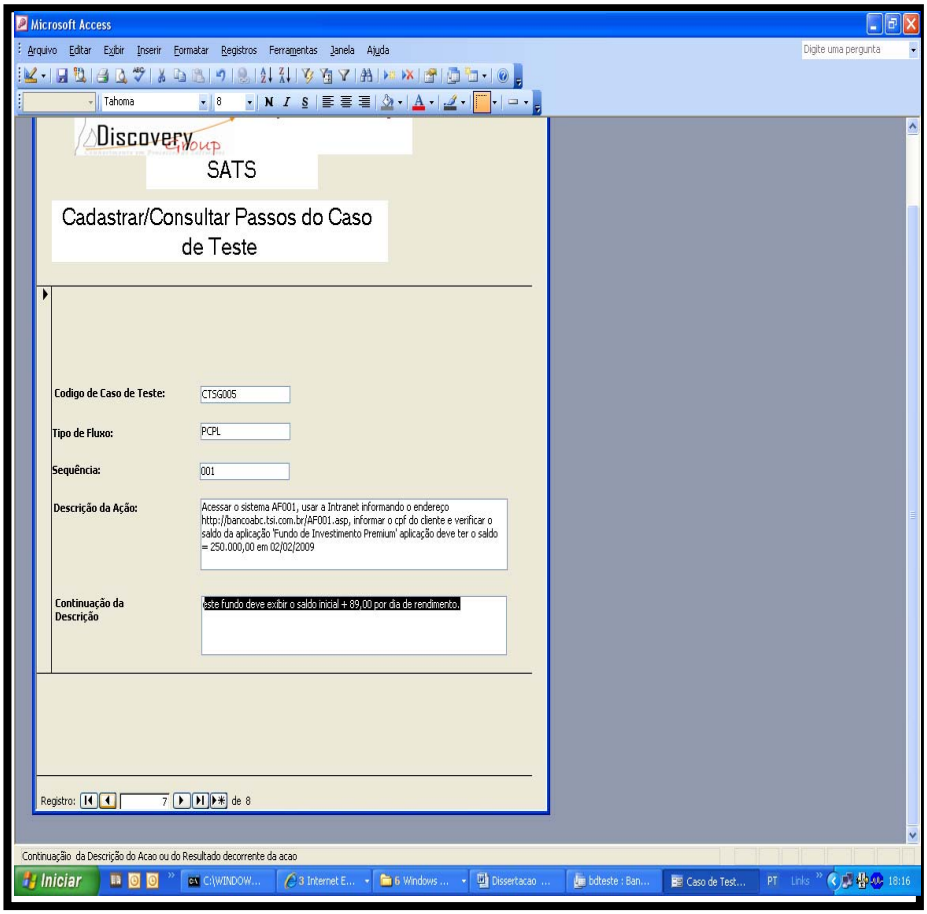


Figura 6.11 – Tela para Cadastramento do passo 2 do caso de teste

Detalhamento do segundo passo com os dados a serem conferidos nas execuções do caso de teste do sistema de Controle de garantia.

Na quarta macroatividade foi feita a execução dos testes, utilizando os casos de testes cadastrados na ferramenta SATS. A utilização da ferramenta auxiliou a comunicação entre os testadores e a equipe de construção do sistema, acelerando a comunicação e o acerto dos artefatos que não se comportavam conforme o previsto. Cada caso de teste com resultado real diferente do esperado era encaminhado à equipe de desenvolvimento, utilizando o correio eletrônico com o a tela do caso de teste anexa e a imagem do teste incorreto. A consulta à ferramenta no módulo de conhecimento e planejamento disponibilizada para toda a equipe facilitou a rápida repetição e entendimento do erro e acerto por parte dos construtores. A figura a seguir ilustra o cadastramento no SATS do resultado incorreto de um caso de teste:

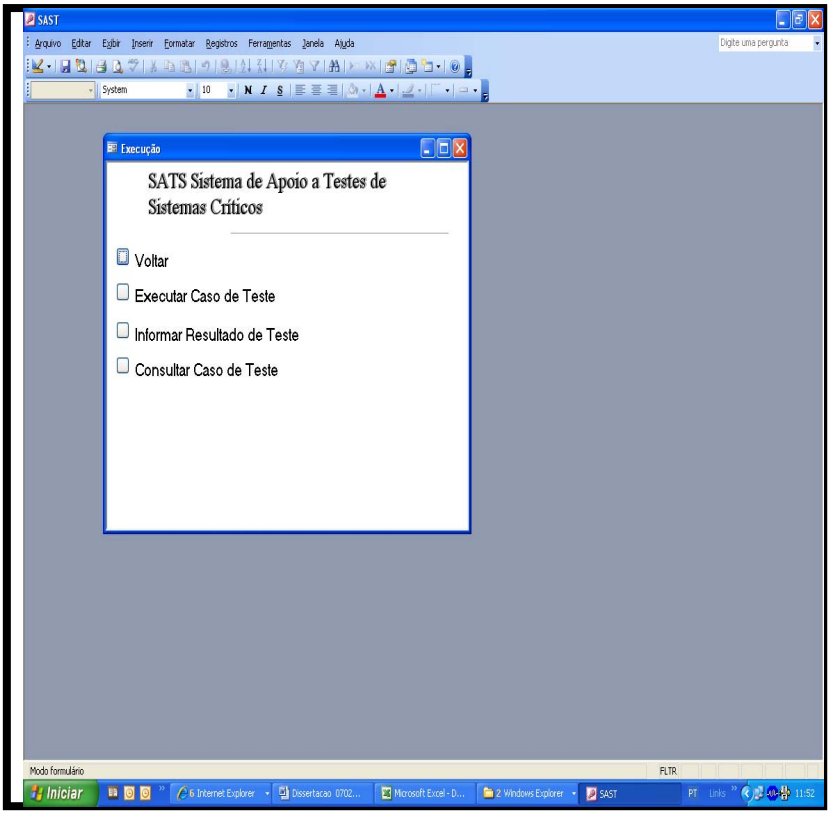


Figura 6.12 – Tela de submenu para informar resultado dos testes executados

Ao ser acionado o botão informar resultado de teste, a tela a seguir é exibida e o testador informa os dados que identificam o caso de teste e aciona o botão passos para informar o resultado do teste do passo testado.

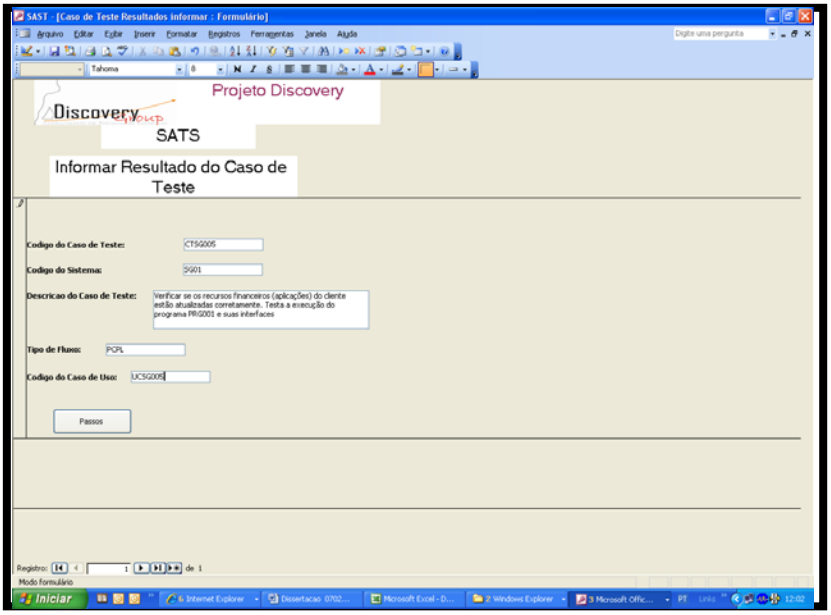


Figura 6.13 – Tela para informação do resultado do teste executado.

No estudo de caso realizado os testes executados e considerados corretos eram imediatamente encaminhados aos homologadores, que podiam revisá-los e homologá-los, ou não, caso detectassem alguma inconsistência ou alteração feita após a confecção dos Casos de Uso e Casos de Teste.

O gerente de testes também teve a possibilidade de acompanhar a evolução da execução dos testes, atuando na prevenção de atrasos nas rotinas pertencentes ao caminho crítico do processo e no acompanhamento da produtividade e precisão da equipe de testes. Os testes, ao serem liberados do ambiente de desenvolvimento, para o ambiente de homologação, tiveram um percentual de erros de 20%. Esse percentual pode ser levantado graças aos dados gravados na ferramenta e foi considerado inferior, pelos usuários, aos dos sistemas anteriormente transferidos. Foi feito levantamento histórico nos registros de teste de quatro sistemas testados anteriormente na empresa. Todos os sistemas tinham extrapolado as horas previstas de testes (Tabela 6.1). Por ocasião da implantação no ambiente de produção online, nenhum problema ocorreu e o sistema foi implantado dentro do prazo previsto, utilizando 50% a menos de recursos de horas/homem na etapa de implantação. A Tabela 6.1 permite melhor visualização:

Sistema	Qtde de Programas	Número de Linhas	Nível de Complexidade	Número de Testadores	Tempo Previsto	Tempo Real	Real/ Previsto
Controle de Recursos Humanos	219	180000	3	6	4000	5000	25%
Central de Risco	297	200000	5	6	2000	3000	50%
Cambio	296	356000	4	6	3200	5000	56%
Convênios	20	16000	3	6	3000	3600	20%
Controle de Garantias	580	464000	7	5	6000	3000	-50%

Tabela 6.1 - Comparativa entre sistemas testados sem e com o uso do SATS

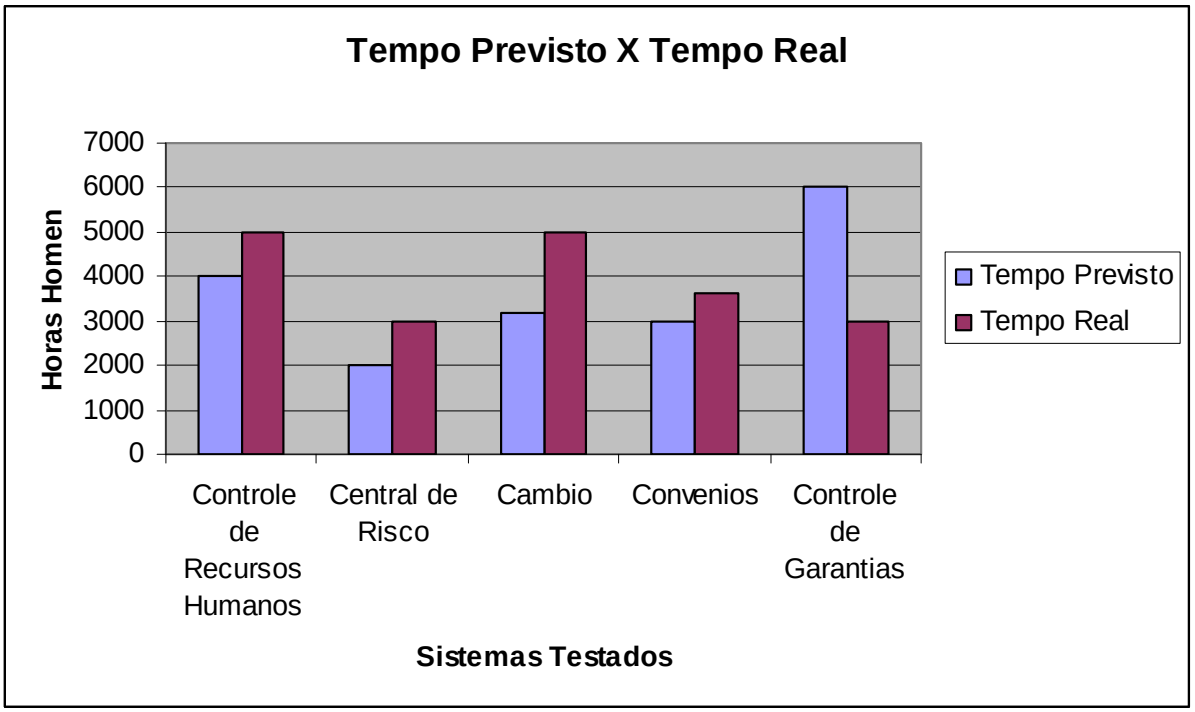


Gráfico 6.1 - Comparativo entre sistemas testados sem e com o uso do SATS-

A equipe de testes utilizada para a execução dos testes era composta de vinte *testers,* cinco analistas de teste, um gerente de projetos certificado PMI e dois auxiliares de gerente.

Após a implantação do sistema, foi distribuído o questionário de satisfação de software (Anexo 4) para ser preenchido pelo usuário responsável pelo sistema e pelos homologadores que acompanharam a conferência e liberação dos artefatos, para que fossem avaliadas as características de qualidade do sistema.

7 CONCLUSÕES

7.1 Considerações Finais

Neste trabalho foi apresentada uma proposta de um modelo de processo para teste de sistemas de software críticos.

Ele está baseado em atividades típicas de processos de teste com planejamento, execução e análise dos resultados e mais a atividade específica para sistemas críticos, que é aquela que identifica as ameaças e riscos e direciona as demais atividades.

O modelo enfatiza também o rigor da execução de todas as atividades e tarefas especificadas haja vista as conseqüências do sistema ir para a produção com falhas não descobertas.

Um estudo de caso de aplicação do processo proposto foi feito em uma instituição financeira de São Paulo que tem a sua área de testes instalada em Belo Horizonte.

A utilização do modelo para teste do Sistema de Controle de Garantias mostrou que a criação de um projeto de teste feito por uma Gerencia de Testes, oficialmente constituída, permite um acompanhamento mais preciso do processo de testes. Os testadores informaram que a execução dos testes feita seguindo os casos de teste cadastrados no SATS, que é a ferramenta que foi desenvolvida para possibilitar o planejamento dos testes e o registro do conhecimento construído na atividade de análise de segurança do sistema, trouxe uma segurança que eles nunca tiveram para liberar os artefatos testados.

Quando os artefatos apresentavam resultados diferentes dos esperados, devido a erros, eles eram facilmente encaminhados para a equipe de desenvolvimento e, depois de corrigidos os erros, a execução de toda a bateria de testes ficou facilitada. Na empresa em que o modelo foi implantado, o relacionamento entre a equipe de análise, programação, testes e homologação melhorou consideravelmente.

Na esfera gerencial e de controle de produtividade, foi possível identificar pontos de gargalo e requisitos que não foram identificados nos levantamentos

iniciais nas interfaces com outros sistemas. A grande vantagem foi a rápida identificação dos requisitos faltosos e a facilidade de seu cadastramento e solução da pendência. A manutenção dos dados de histórico de execução dos testes permitiu a empresa criar métricas individualizadas por sistemas, equipes e até atividades.

O modelo de testes já foi implantado em uma instituição financeira e uma fábrica de software em Belo Horizonte e algumas alterações já foram solicitadas e realizadas de forma a atender a realidade das empresas.

7.2 Contribuições Principais

Dentre as contribuições deste trabalho podemos destacar como principais:

• A definição de um modelo de processo de testes para softwares de sistemas críticos, baseado na literatura e aderente à norma ISO/IEC 12207 (ISO/IEC, 1998; ISO/IEC, 2002; ISO/IEC, 2004) e aos modelos CMMI (SEI, 2002), MPS.BR (MPS.BR, 2009) e MPT.BR (MPT.BR,2009)

• A definição de uma estratégia de integração das atividades de testes de sistemas críticos ao processo de desenvolvimento de software;

• A definição de um conjunto inicial de critérios para a classificação das ameaças e riscos a que o sistema crítico em questão está exposto. Esse conjunto foi definido a partir de uma seleção da literatura existente, baseada na probabilidade de ocorrência da ameaça e possíveis conseqüência do evento. Vale salientar que esses critérios representam um conjunto que, apesar de ser baseado na literatura, ainda não foi avaliado através de experimentos formais. Apesar disso, ele já está em uso na indústria, o que possibilitará sua avaliação formal e identificação de possíveis adequações e melhorias; e

• A definição e implementação de uma ferramenta para planejar, controlar e auxiliar a execução das atividades de testes.

Apesar da implementação do modelo proposto já estar acontecendo em duas empresas, os benefícios dessa abordagem só poderão ser efetivamente avaliados em um procedimento de avaliação formal e, depois de um tempo, experimentando a proposta. Porém, como a validação do modelo implica na sua utilização em vários

projetos e excede, em muito, o tempo esperado para o desenvolvimento de uma dissertação de mestrado, a validação da abordagem proposta poderá ser realizada no contexto do Projeto Discovery. Inclui-se, entretanto, no capítulo 6, um exemplo de sua utilização.

7.3 Trabalhos Futuros

Buscando melhorar e expandir o Modelo de Testes Para Sistemas Críticos Proposto, algumas perspectivas de trabalhos futuros são destacadas.

Uma perspectiva futura interessante seria a de caracterizar a aplicabilidade do Modelo proposto neste trabalho, na prática, através do uso sistemático por empresas.

No estudo de caso de aplicação do processo proposto foi identificada a carência de empresas com habilidade e conhecimento para fazer a homologação de sistemas críticos e ao mesmo tempo uma demanda das empresas construtoras de software por esse serviço.

Pode ser feito a complementação da ferramenta criando as consultas para acompanhamento de prazos previstos X prazos reais, com o objetivo de conceber métricas relacionando a experiência das equipes com o nível de acerto dos prazos previstos.

Outro desdobramento possível deste trabalho seria planejar e executar um estudo experimental para avaliar o impacto nos custos de desenvolvimento e produção de sistemas críticos que utilizaram e que não utilizaram o modelo de testes sugerido neste trabalho.

Outra perspectiva futura seria a utilização do mecanismo de *Storytelling* para guardar e disseminar conhecimentos referentes às ameaças, riscos, fatores de criticidade dos sistemas críticos, incrementando o modelo, principalmente, na de execução dos testes.

Por fim, com relação à incorporação desta pesquisa ao Projeto Discovery, é relevante investir na forma de exibição visual do conhecimento. Seria útil que os dados referentes ao controle e acompanhamento dos testes de sistemas críticos pudessem ser visualizados graficamente de forma a facilitar o entendimento e o

controle gerencial do andamento do processo de teste de software. A situação real do projeto de teste, a nível global ou em nível de artefato, poderia ser visualizada usando recursos como o glifo borboleta proposto por Guedes (2007), por exemplo.

REFERÊNCIAS

ALATS ASSOCIAÇÃO LATINO AMERICANA DE TESTES DE SOFTWARE (ALATS). Disponível em:<www.alats.gov.>Acesso em: 01 de jun. 2009.

ALMEIDA JUNIOR, Jorge. R. Segurança em sistemas críticos e em sistemas de Informação: um estudo comparativo. 2003. 191 f. Tese de Livre Docência. Escola Politécnica da USP, São Paulo: São Paulo. Disponível em:<www.cos.ufrj.br/~taba> Acesso em:<10 de nov. 2008.

Avizienis, A; Laprie J.; Randell B; Landwehr C; Basic Concepts and Taxonomy of Dependable and Secure Computing, Technical Research Report, Institute for systems Research, 2004 http://www.isr.umd.edu

BEIZER, Borris. Black-Box Testing: techniques for funcional testing of software and systems. Wiley, New York, 1995.

BERKELEY - SCHOOL OF INFORMATION MANAGEMENT & SYSTEMS,UNIVERSITY OF CALIFORNIA. How Much Information? Executive Summary. Apresenta informações estatísticas sobre a quantidade de informações existentes no mundo. Disponível em: <http://sims.berkeley.edu/resea

BOEHM, Barry W. Software Engineering Economics. Englewood Cliffs, NJ Prentice-Hall, 1981 ISBN 0-13-822122-7.

BOEHM, Barry W. A Spiral Model of Software Development and Enhancement. A Spiral IEEE Computer, v.21, n.2, p. 61-72 ,1988.

Burnstein, I. Practical Software Testing, Springer Professional Computing, 2002, ISBN 0-387-95131-8

CARVALHO, André C. P. L. F. *et. al* (2006) Grandes Desafios da Pesquisa em Computação no Brasil . SBC Sociedade Brasileira de Computação, maio/2006, São Paulo-Campinas, 2006. p.21.

CHRISSIS, M. B., Konrad, M., S. CMMI: Guidelines for Process Integration and Product Improvement. Addison-Wesley, 2003.

CRESPO, Adalberto N. Uma metodologia para teste de software no contexto da melhoria de processos. Campinas: CenPRA-Centro de Pesquisa Renato Archer. SBQS – Simpósio Brasileiro de Qualidade de Software, 2004.

CRUZ, Jorge Suporte automatizado à rastreabilidade em um processo de teste de software baseado em documentação...Anais.. Vila Velha: Simpósio Brasileiro de Qualidade de Software, 5, pp. 278-292, 2006.

DEMAIC disponível em http://store.isixsigma.com/product.asp?P_ID=220, acessado dia 14/05/2009

DIAS, Cláudia Disponível em: http://www.devmedia.com.br/resumo/default.asp?ed=6&site=48. Acesso em: 20 de ago. 2009.

DIAS Cláudia. Segurança e auditoria da Tecnologia da Informação. Rio de Janeiro: Axcel Books, 2000. 218p. Disponível em:< www.softex.br. >Acesso em: 11 de out. 2009.

ESPINHA Rafael ; Melhorando Processos Através da Análise de Risco e Conformidade artigo revista Engenharia de Software Magazine disponível em http://www.devmedia.com.br/articles/viewcomp.asp?comp=8030&hl=, acessado 15/10/2009

ESTOLANO Mário. H. R. Base de Métricas para a Estação TABA. 2005. 125 **f.** Dissertação de Mestrado. COPPE/UFRJ. Rio de Janeiro

FEDERAL HIGHWAY ADMINISTRATION (FHWA) Clarus Concept of Operations. Publication No. FHWA-JPO-05-072,

GUEDES, Fabiana C. Proposta de visualização da verificação de processos de desenvolvimento de software por meio da rastreabilidade.Dissertação de mestrado 188f. Belo Horizonte: PUCMINAS, 2007.

GOUVÊIA, Ana Paula O. Modelo de Melhoria Contínua do Processo de Teste a partir da Validação do Produto de Software**.** Dissertação de mestrado 178 f. Belo Horizonte: PUCMINAS, 2008.

HETZEL, Bill. The Complete Guide to Software Testing**.** Second Edition, John Wiley & Sons, 1988.

HSE, 2009. Disponível em:<www http://www.atl.org.uk/health-and-safety/legal-framework/health-safety-legislation.asp.>Acesso em: 01 de jul. 2009.

HOLLNAGEL, E.; WOODS, D.; LEVESON, N. International Symposium on Resilience Engineering, 2004. Disponível em:<http://csel.eng.ohio-state.edu/woods/error/About%20Resilience%20Engineer.pdf.>Acesso em:09 de mar. 2006.

HUMPHREY, Watts. Managing the Software Process. Addison-Wesley Professional. ISBN 0-201-18095-2, 2006.

IEC INTERNATIONAL ELETROTECHNICAL COMMISSION. FUNCTIONAL SAFETY OF ELECTRICAL/ELECTRONIC/PROGRAMMABLE ELECTRONIC SAFETY-RELATED SYSTEMS, Part1 to 7, version 3.0, IEC 61508. 1997.

INTERNATIONAL ATOMIC ENERGY AGENCY. ANUAL (UPEA) Report 2001, GC (46)/2. 2002, Viena, Áustria. 162p.

INTERNATIONAL ATOMIC ENERGY AGENCY. INSTRUMENTATION AND CONTROL SYSTEMS IMPORTANT TO SAFETY IN NUCLEAR POWER PLANTS. Safety Standard Series, No. NS-6-1.3,2002b. 91p.

INTERNATIONAL ATOMIC ENERGY AGENCY. Disponível em:<www.iaea.org acesso em 01/07/2009.

ISO 8402 ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS – Gestão da qualidade e garantia da qualidade – Terminologia. Rio de Janeiro: ABNT, 1994.

ISO/IEC 9126 ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. Engenharia de software: Qualidade de produto. Parte 1: Modelo de qualidade, 2003.

ISO/IEC 12207, INTERNATIONAL ORGANIZATION FOR STANDARDIZATION/ INTERNATIONAL ELECTROTECHNICAL COMISSION. Systems and software engineering– Software life cycle processes, Geneve: ISO, 2008.

ISO/IEC 15504, ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS Technology Information – IT Service Management Process Assessment Model educatation. 2007.Disponível em:<www.isospice.com/> Acesso em: 01 de jul de 2009.

IPPOLITO, Lourdes. M.; WALLACE, Dolores. R**.** A Study on Hazard Analysis in High Integrity software Standards and Guidelines. National Institute of Standards and Technology.2009. Disponível em:<http://hissa.ncsl.nist.gov/ HHRFdata/Artifacts/ITLdoc/558>Acesso em 3 de jul. 2009.

KANER, Cem. Bach, Bach. Pettichord, Bret. Lessons Learned in Software Testing: A Context Driven Approach, Wiley, 2002

KUHN, Richard D. Reilly, Michael J.An Investigation of the Applicability of Design Experiments to Software Testing, 27th NASA/IEEE Software Engineering Work-shop, NASA Goddard Space Flight Center, 4-6 December 2002. disponível em http://csrc.nist.gov/staff/kuhn/kuhn-reilly-02.pdf

KUVAJA, P. Software Process Assessment and Improvement: The BOOTSTRAP Approach, Blackwell. Publishers, Oxford, 1994.

LEVESON, N. G. , HARVEY P. R. Analyzing software Safety. IEEE Transactions on software Engineering , Setembro de 1983, Vol. SE-9, nº 5.

LEVESON, N. G. Safeware systems safety and computers. New York: Addison Wesley Publishing Company, 1995.

LEVESON, N. G. Software Safety in Embedded Computer Systems. Comunications of the ACM archive, vol.34, Issue 2, February 1991, Full text, pdf2,83 MB)

LOVISI FILHO, Elio.; Cunha, Adilson Marques da. Uma Proposta De Sistemática Para A Análise De Segurança De Software Crítico Embarcado Aeroespacial... Anais...Lisboa: 4º Encontro de Qualidade nas Tecnologias da Informação, 2001.

MARICK, Brian. Classic Testing Mistakes 1997. Disponível em:<http://www.visibleworkings.com/papers/mistakes.pdf .>Acesso em: 01 de jul. 2009.

MSF - MICROSOFT SOLUTIONS FRAMEWORK . Disponível em: http://technet.microsoft.com/pt-br/library/bb490188(printer).aspx acesso em 12 de out. 2008.

MINISTÉRIO DA CIÊNCIA E TECNOLOGIA, SECRETARIA DE POLÍTICA DE INFORMÁTICA (MTC). Qualidade e Produtividade no Setor de Software Brasileiro. Brasília:2001.

MINISTRY OF DEFENSE ; Standard Pratice for a System Safety MIL-STD-882D 10, February , 2000.

MOLINARI, Leonardo: Testes de Softwares: produzindo sistemas melhores e mais confiáveis. São Paulo: Érica, 2003.

MONTONI, M. *et. al*. MPS Model and TABA Workstation: implementing a software Process Improvement Initiatives in Small Settings.

MOURA C. A. T. Uma estratégia de análise de segurança de software para aplicações Críticas**.** 1996, 208 f.Tese de Doutorado S. José dos Campos: ITA, 1996.

MPS.BR – Guia de Implementação . Parte 1: fundamentação para implementação do nível G até o nível A do MR-MPS:2009, maio 2009. Disponível em: www.softex.b Acesso em 5 de mai. 2009.r.

MPT.BR. Guia de implementação.Melhoria do Processo de Teste Brasileiro. Parte 1:Nível 1 versão 1.3.6. Disponível em:<www.alats.gov. acessado em 01/06/2009.

MYERS, Glenford J. The Art of Software Testing. New York: Wiley, 1979.

NCOSE, Forsberg, K. and Mooz, H., The Relationship of Systems Engineering to the Project Cycle, First Annual Symposium of the National Council On Systems Engineering, October 1991.

NORMA IEEE STD 829 STANDARD FOR SOFTWARE DOCUMENTATION.Revisão da IEEE Std 829-1983, 1988.

PAULA FILHO, Wilson de Pádua. Engenharia de Software Magazine edição especial. Disponível em:< www.devmedia.com.br/esm acessado 28/02/2009.>Acesso em:01 de jul de 2009.

PFLEEGER ,Shari L. Engenharia de Software, 2004.

PHADKE Madhav S.“Design Of Experiment for Software Testing, Janeiro 2003, disponível em http://www.isixsigma.com/library/content/c030106a.asp

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS. Pró-Reitoria de Graduação. Sistema de Bibliotecas. Padrão PUC Minas de normalização: normas da ABNT para apresentação de trabalhos científicos, teses, dissertações e monografias. Belo Horizonte, 2008. Disponível em:<http://www.pucminas.br/biblioteca.>Acesso em: 20 de jul. 2009.

PÔRTO I. J., DE BORTOLI L. A. Sistemas tolerantes a falhas. Disponível em:<htpp//pesquisa/tf/portugues/ensino/lisangela/segsoft. html, .>Acesso em: 20Jul. de 2008.

RIOS, E.; MOREIRA FILHO, T. Teste de Software. São Paulo: AltaBooks, 2002.

RODRIGUES D. P., Software Safety Verification Critical Software Intensive Systems. Eindhoven. NY: Universiteit Eindhoven, 2002 ISBN 90-386-0953-0.

SAEED, A.; LEMOS, R.; ANDERSON, T. The role of formal methods in the requirements analysis of safety-critical systems: a train set example. In: IEEE INTERNATIONAL SYMPOSIUM ON FAULT TOLERANT COMPUTING, 2, Montreal, Canadá, 1991. p.478-85.

SANDEEP MAHER, SR. TEST MANAGER. Disponível em:<www.testrepublic.com> Acesso em 18 de mar. 2009.

SANDHOF, Karen; FILGUEIRAS , Lucia Vilela Leite. Defeitos de Softwares como erros humanos....Anais... São Paulo:, II Workshop Um Olhar Sociotécnico sobre a Engenharia de Software – WOSES, 2007.

SANTELLANO J. , MOURA C. A. T. , LIMA A. C. Estratégias de Segurança de software: A Abordagem do Padrão MIL-STD-498 , 1998.

SOFTEX. ASSOCIAÇÃO PARA PROMOÇÃO DA EXCELÊNCIA DO SOFTWARE BRASILEIRO MPS.BR – Guia Geral, v 1.1, 2009. Disponível em:<www.softex.br> Acesso em: 5 nov. 2009.

SOMMERVILLE, Ian. Engenharia de Software 8ª. edição,São Paulo,Pearson Edison Wesley, 2008.

SOUZA, R. V. G. Caracteristicas de testabilidade nos diagramas ULM (Unified Modeling Language): apoio aos testes de sistemas de software orientados a objetos. 188 f. Tese de Doutorado. São Paulo: USP, 2003.

SPICE SOFTWARE PROCESS IMPROVEMENT AND CAPABILITY DETERMINATION ISO/IEC 15504, 20003 Disponível em http://www.isospice.com/categories/ISO%7B47%7DIEC-15504-Standard/ acessado em 02/12/2009

SPILLNER, Andreas. From V-model to W-model - Establishing the Whole Test Process Conquest 2000, Workshop on »Testing Non-Functional Software-Requirements. Nuremberg, Germany, 13.-15. September 2000, Proceedings, ASQF e.V., 2000, pp 222-231 (invited talk)

STOREY,N. Safety-Critical Computer Systems. New York: Addison Wesley, 1996. 453p.

THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS. IEEE Std 829: Standard for Software Test Documentation. New York: IEEE Computer Society, September, 1998

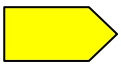
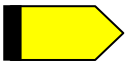
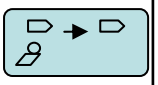
TMMi TEST MATURITY MODEL INTEGRATION. Disponível em:<www.tmmifoundation.org/> Acesso em: 12 de dez 2007.

UALG, 2009. Disponível em:< http://w3.ualg.pt/~pventura/ep/aulas_tp/t1_g5.pdf.>Acesso em: 5 de jul 2009.

VERGARA, Silva C. Projetos e relatórios de Pesquisa em Administração. 4ª ed. São Paulo: Atlas, 2003. 93p.

WALLACE, Dolores. R. ; IPPOLITO, Lourdes M. A framework for the development and assurance of high integrity software. National institute of Standards and technology. NIST Special Publication 500-223. December 1994

ANEXO 1 NOTAÇÃO UTILIZADA NA MODELAGEM DA PROPOSTA DE TESTES DE SISTEMAS CRÍTICOS

Objeto	Notação	Definição
Ator		Objeto que representa uma pessoa, agente ou unidade organizacional
Atividade		Objeto que se refere ao conceito de atividade
Atividade composta		Objeto que se refere ao conceito de atividade composta e que pode ser dividida em subatividades.
Estado Inicial		Objeto puramente notacional, proveniente dos diagramas de estado e que indica onde é iniciado o fluxo de atividades que definem um processo ou uma atividade composta
Estado Final		Objeto puramente notacional, proveniente dos diagramas de estado e que indica onde é finalizado o fluxo de atividades que definem um processo ou uma atividade composta
Documento		Objeto referente a um documento
Operação Lógica OR		Operação Lógica OU / OR
Junção após operação lógica		Notação sem adorno usada como elemento de junção
Área de ator		Área que agrupa atividades exercidas por um ator ou grupo de atores. O ator ou o grupo de atores deve estqr contido na área
Fluxo de entrada e Saída		Ligação que estabelece um insumo (se o fluxo é de entrada) ou um produto de uma atividade

ANEXO 2 INVENTÁRIO DE AMEAÇAS

Nome do Projeto			**SSI Número**		
Analista de Negócio					
Elaborado por		**Em**			
Revisado por		**Em**		**Versão**	

#	Descrição	Área de Impacto	Gravidade (1 – 5)	Data de identificação	Ação Requerida (Risk Mitigation)	Data Limite	Responsável pela ação	Data da eliminação
1								
2								
3								
4								
5								
6								
7								
8								
9								
10								
...	...		...	...	...	...	...	...

<# Número seqüencial Da ameaça/risco identificado<
< DESCRIÇÃO: Descrever a ameaça/perigo/ risco encontrado >
< ÁREA DE IMPACTO: Determine a área que o risco encontrado terá mais foco no escopo do projeto, como exemplo: FINANCEIRA, CONTÁBIL, NEGÓCIO, PESSOAL, ETC.
< GRAVIDADE: Mensurar a gravidade do risco no desenvolvimento do projeto. >
< DATA DE IDENTIFICAÇÃO: Data em que foi detectado o risco. >
< AÇÃO REQUERIDA: Descreva a ação requerida para eliminação do risco. >
< DATA LIMITE: Data limite para eliminação do risco >
< RESPONSÁVEL PELA AÇÃO: Determine o ator responsável pela eliminação do risco encontrado. Contextualize a identificação em relação ao grupo envolvido no projeto utilizando, como exemplo, o nome e sobrenome do ator. >
< DATA DA ELIMINAÇÃO: Data da eliminação do risco >

ANEXO 3 PADRÕES PARA ELEMENTOS DE PROGRAMAÇÃO DO BANCO ABC

ÍNDICE

INTRODUÇÃO

O propósito deste documento é apresentar o escopo da documentação necessária para implementação de um padrão de nomenclatura para os chamados objetos sistêmicos atendendo ao desenvolvimento de sistemas da ALTA e BAIXA plataforma. Há objetos específicos de cada plataforma os quais estarão descritos nos tópicos relativos a plataforma em questão, todos os demais objetos serão comuns as plataformas.

OBJETIVOS

- Identificar todos os Objetos Sistêmicos existentes na empresa;
- Descrever sucintamente cada tipo de Objeto Sistêmico;
- Padronizar e normatizar os Objetos Sistêmicos facilitando a aquisição do conhecimento a construção e manutenção dos programas criados para os sistemas críticos.
-

BENEFÍCIO

- Simplifica a visualização e identificação de Objetos Sistêmicos nos vários ambientes operacionais da empresa;
- Individualiza cada um dos Objetos Sistêmicos utilizados na empresa;
- Identifica uma única regra, padronizada e difundida para formação de códigos de objetos.
- Permite a rastreabilidade dos Fatores de criticidade tratados internamente nos objetos sistêmicos

DEFINIÇÃO

Objeto sistêmico é todo e qualquer objeto que é necessário no desenvolvimento e implantação de um projeto/sistema.

COMPOSIÇÃO

A Regra Geral para formatação dos NOMES DOS OBJETOS SISTEMICOS será a definida abaixo:

Sistema		Objeto	Variação				
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**

Onde:

SISTEMA: Identificador do Sistema, 2 caracteres alfabéticos, por exemplo: IP – Informações Pessoais de Cliente.

OBS.: O mnemônico para o nome do Sistema não deve iniciar com **C**, pois nesta posição ela e reservada ao CICS.

OBJETO: Identificador do tipo de objeto sistêmico (1 caracter alfabético). Este caracter alfabético respeitará a regra para cada objeto sistêmico em questão.

VARIAÇÃO: Será a composição de um item alfanumérico responsável pela especificação do Objeto Sistêmico (5 caracteres alfanuméricos). Estes 5 caracteres respeitarão a regra de cada objeto sistêmico em questão.
Exemplo : IPB00001 (Programa Batch de número UM do sistema de Informações Pessoais de Cliente).

DOCUMENTAÇÃO

A regra que definirá a formatação para os artefatos de documentação que serão os TEMPLATES e OUTROS documentos que se fizerem necessários será:

Sistema		Objeto	Variação				Sufixo	Hifen	Livre
1	2	3	4	5	6	7	8	-	(118)

Onde:
SISTEMA: Identificador do Sistema, 2 caracteres alfabéticos.

OBJETO: Fixo para este objeto sistêmico em **T**.

VARIAÇÃO: Nas duas primeiras posições (4 e 5) coloque os dois últimos números que identificam o Template na fase da Metodologia de Desenvolvimento do BANCO ABC. Para outros documentos que não estejam previstos na Metodologia utilize o número **99**.
Exemplo: Para o Template 3.**08** PF Especificação do Caso de uso o número seria **08**.
Para as duas últimas posições da Variação (6 e 7) utilize o número seqüencial para o Template ou o Documento em questão, 01, 02, 03, 04, 05 etc.

SUFIXO: No sufixo especifique a fase da Metodologia de Desenvolvimento do BANCO ABC a que se refere o Template, de acordo com a tabela abaixo. Para outros documentos que não estejam previstos na Metodologia utilize a letra **O**.

Tipos de Sufixo possíveis:
A : Para Templates da fase de ANTE-PROJETO.
L : Para Templates da fase de PROJETO LÓGICO.
F : Para Templates da fase de PROJETO FÍSICO.
C : Para Templates da fase de CODIFICAÇÃO.
T : Para Templates da fase de TESTES INTEGRADOS.
H : Para Templates da fase de HOMOLOGAÇÃO.
I : Para Templates da fase de IMPLANTAÇÃO.
G : Para Templates de GESTÃO.
O : Para OUTROS documentos.

Exemplo : IPT0801F (Primeiro Template, 3.08 PF Especificação do Caso de uso para o Sistema de Informações Pessoais de Cliente).

HÍFEN: Fixo, esta posição receberá um hífen.

LIVRE: Nesta posição você terá 118 caracteres livres para melhor descrever o documento.
Exemplo : IPT0801F-Especificação do Caso de uso para Cadastro de Informações Pessoais de Cliente.

ARQUIVOS DE DADOS

A regra que definirá a formatação para os nomes dos ARQUIVOS DE DADOS que não sejam de um Banco de Dados específico, como exemplo: VSAM, QSAM, Arquivos texto, etc, será:

Sistema		Objeto	Seqüencial			Sufixo	Identificador
1	2	3	4	5	6	7	8

SISTEMA: Identificador do Sistema, 2 caracteres alfabéticos.

OBJETO: Fixo para este objeto sistêmico em **D**.

SEQUENCIAL: Seqüencial numérico.

SUFIXO: Sufixos disponíveis.

BAIXA PLATAFORMA: Fixo em **Z**;

ALTA PLATAFORMA: *Sufixos disponíveis para o **DDNAME***:

V : Identificador de arquivos VSAM (MVS) para INPUT;
I : Identificador de arquivos seqüenciais (MVS) para INPUT;
S : Identificador de arquivos VSAM (MVS) para OUTPUT;
O : Identificador de arquivos seqüenciais (MVS) para OUTPUT;
P : Identificador de arquivos seqüenciais particionados (MVS);
U : Identificador de arquivos VSAM (MVS) para UPDATE;
E : Identificador de arquivos seqüenciais (MVS) para EXTENDED;

IDENTIFICADOR : Identificador do Arquivo, conforme o programa:
(1..9) : Ordem de execução dos arquivos conforme o sufixo;

DSNAME

De forma especial, **teremos** para o **DSN** :
ABCxx.ppscmeee.AAD999.aaaaaaaa.bbbbbbb.cccccccc,
onde:

xx é o identificador de localização do arquivo :
PD – Produção Disco
PS – Produção Software
PC – Produção Cartucho
DD – Desenvolvimento Disco

DS – Desenvolvimento Software
DC – Desenvolvimento Cartucho
SI – Suporte
SS – Suporte Software
SC – Suporte Cartucho

pp é o identificador do profile RACF :
A0 – todos alteram
A1 – todos **lêem**, G1, G2, G3 e usuários finais alteram
A2 – todos lêem, G1, G2 e G3 alteram
A3 –
A4 – todos lêem, G1 altera e G2 e G3 controlam
A5 – todos lêem, G1 altera
A6 – ninguém lê, G1 altera e G4 controla
A7 – ninguém lê, G1 altera e G2, G3 e G4 controlam
A8 – ninguém lê, G1 altera e G2 controla
A9 – ninguém lê, G1 altera
Onde, G1 – Área de Suporte e Análise de Produção
G2 – Produção Supervisão Preparação e Supervisão Controle
G3 – Produção Preparação e Controle
G4 – Funcionários do Banco

s é o identificador do SG/SC :
R – residente
S – spool
T – temporário
W – work
C – CICS

c é o identificador do DC :
0 – prevalece informações do JCL
1 – compress
2 – PDSE
3 – PDS

m é o identificador do MC :
0 – MIGRATE=N
1 – MIGRATE=Y, PRI=1, ML1=1, ML2=28
2 – MIGRATE=Y, PRI=7, ML1=0, ML2=23
3 – MIGRATE=Y, PRI=5, ML1=5, ML2=0
4 – MIGRATE=Y, PRI=5, ML1=0, ML2=15

eee é o código das empresas :
001 – Multi-Empresa
002 – Banco ABC
020 – Banco Arabian

022 – Banco Unico

AAD999 é o DDNAME do arquivo (definido acima);

aaaaaaaa é mnemônico identificador do arquivo;
bbbbbbbb é mnemônico identificador do código da PROC **(se houver);**
ccccccccc é o mnemônico identificar do SORT ou UNLOAD **(se houver)**.

Obs.: a geração de GDG segue o padrão já adotado hoje.

RELATÓRIO

A regra que definirá a formatação para os nomes dos programas de RELATÓRIOS será:

Sistema		Objeto	Seqüencial				
1	2	3	4	5	6	7	8

Onde:
SISTEMA: Identificador do Sistema, 2 caracteres alfabéticos.

OBJETO: Fixo para este objeto sistêmico em **L**.

SEQUENCIA*L*: Seqüencial numérico.

RELATÓRIOS (Cabeçalhos)

O cabeçalho dos relatórios deverá conter os seguintes objetos opcionais e obrigatórios:

Objeto	Label	Descrição
LOGOMARCA OU NOME DA EMPRESA		É a logomarca ou o nome da empresa para o qual o relatório foi criado.
NOME DO SISTEMA		É o nome do sistema, mas não é necessária a sigla, pois o nome do programa já a contemplará.
DATA DE PROCESSAMENTO	DATA PRO	Será a data e hora em que foi processado e impresso o relatório.
HORA DE PROCESSAMENTO	HORA PRO	Será a hora em que foi processado e impresso o relatório.
DATA REFÊRENCIA	DATA REF	É a data a que se refere o conteúdo dos dados impressos no relatório.
NOME DO RELATÓRIO		É o título do relatório, nome que o relatório deverá ter. Exemplo: POSIÇÃO DA CARTEIRA DE FUNDOS.
CÓDIGO DA EMPRESA		É o código de 4 posições pelo qual a empresa é referenciada no i back-end (framework). IMPORTANTE: OBRIGATÓRIAMENTE o código da empresa deverá iniciar na LINHA 03 a partir da COLUNA 01.
QUEBRA GRUPO 1 QUEBRA GRUPO 2 QUEBRA GRUPO N (...)		São as quebras que o relatório terá. IMPORTANTE: 1) OBRIGATÓRIAMENTE todas as quebras do relatório acontecerão na LINHA 03, a primeira quebra (quebra do grupo 1) deverá iniciar na LINHA 03 , após o CODIGO DA EMPRESA, a partir da COLUNA 06 de todas as páginas impressas. 2) Todas as quebras serão compostas por um TÍTULO (em maiúsculo), dois pontos, um espaço em branco e a parte variável. 3) Separe uma quebra da outra com dois espaços em branco. 4) Os TÍTULOS para as quebras são PADRONIZADOS, se não encontrar o titulo desejado na lista abaixo contacte o GESTOR DO PROJETO para que o mesmo lhe forneça um TITULO válido homologado junto a AREA DE PRODUÇÃO. LISTA DE TÍTULOS: REGIONAL, AGENCIA, FILIAL, CNPJ, CGC, PAB, SUCURSAL. No exemplo abaixo o relatório terá duas quebras, REGIONAL e AGENCIA, assim na LINHA 03 as quebras ficarão desta forma:

		1234 REGIONAL: I AGENCIA: 0002 (onde 1234 é o CODIGO DA EMPRESA).
NOME DO PROGRAMA DE IMPRESSÃO	XXXXXXXX	É o nome de 8 posições do programa que efetivamente imprime o relatório.
PAGINA	PAG	Número da pagina impressa.
NOME DO PROGRAMA DE PROCESSAMENTO DA IMPRESSÃO	YYYYYYYY	É o nome de 8 posições do programa que processa o relatório.

CONSIDERAÇÕES:

1. O nome do sistema, nome do relatório e quebra principal possuem alinhamento central.
2. Para os campos de taxas e percentuais, utilizar, no relatório, o símbolo apropriado (%).
3. Utilizar nas datas o ano com quatro posições.
4. Após as linhas do cabeçalho virá a **LINHA DE DIVISÃO**, que será a divisão do cabeçalho e o corpo do relatório podendo ser uma linha continua ou traço, use o que melhor convier.
5. Após a LINHA DE DIVISÃO virá a **LINHA DE CABEÇALHO DAS COLUNAS**, com os títulos de cada coluna.
6. Após a LINHA DE CABEÇALHO DAS COLUNAS, virá outra LINHA DE DIVISÃO.
7. Após a última linha acima imprima os dados.
8. Relatórios impressos no padrão **TEXTO** ou **CARACTER** terão obrigatoriamente **132 COLUNAS**.
9. Linhas de **TOTAIS** ou **SOMATÓRIOS** virão ao final da quebra ou relatório com os seus **TÍTULOS** ajustados a direita.
10. O **TIPO DE LETRA**, quando for o caso, para impressão do relatório será **ARIAL**.

Para o desenho do layout do cabeçalho dos relatórios será seguida a regra abaixo, tanto para ferramentas de desenho visuais (crystal reports ou reporting service) ou relatórios formatados para caracter ou texto.

```
12345678901234567890123456789012345678901234567890123456789012345678901234567890123
456789012345678901234567890123456789012345678901234567890
01 (CÓDIGO DA EMPRESA) - LOGO OU NOME DA EMPRESA                          NOME DO
SISTEMA                                DATA PRO:00/00/0000
02 DATA REF:00/00/0000                                                   NOME DO
RELÁTORIO                                 HORA PRO:00:00:00
03 AG.:9999 - XXXXXXXXXX(QUEBRA Grupo 1) (QUEBRA Grupo 2) (QUEBRA Grupo N)
(...)YYYYYY XXXXXXXX – PAG:000
-------------------------------------------------------------------------------
04 NOME DA COLUNA         NOME DA COLUNA          NOME DA COLUNA           NOME DA
COLUNA         NOME DA COLUNA           NOME DA COLUNA
-------------------------------------------------------------------------------
05 XXXXXXXXXXXXXX         XXXXXXXXXXXXXX          XXXXXXXXXXXXXX     XXXXXXXXXXXXXX
XXXXXXXXXXXXXX              999.999,99
06 XXXXXXXXXXXXXX         XXXXXXXXXXXXXX          XXXXXXXXXXXXXX     XXXXXXXXXXXXXX
XXXXXXXXXXXXXX              999.999,99

TOTAL:9.999.999,99
```

A partir deste ponto o documento tratará distintamente os objetos específicos das plataformas alta e baixa, todos os padrões acima são comuns as duas plataformas.

PLATAFORMA ALTA

JOB

JOB: Conjuntos de comandos de JCL que serão submetidos ao Sistema Operacional MVS para execução de um ou mais programas batch.
A regra que definirá a formatação para os nomes dos JOB'S será:

Sistema		Objeto	Seqüencial			Periodicidade	Tipo
1	2	3	4	5	6	7	8

SISTEMA: Identificador do Sistema, 2 caracteres alfabéticos.

OBJETO: Fixo para este objeto sistêmico em J.

SEQUENCIAL: Seqüencial alfa-numérico.

PERIODICIDADE: Tempo/Característica de execução do JOB.

D : JOB Diário
S : JOB Semanal
Q : JOB Quinzenal
M : JOB Mensal
B : JOB Bimestral
T : JOB Trimestral
U : JOB Quadrimestral
W : JOB Semestral
A : JOB Anual
P : JOB (A pedido)

TIPO: Determina a característica do JOB.

O : JOB de erro
R : JOB de Recuperação
K : JOB de Transmissão
Z : nenhum dos tipos acima
N : JOB iniciado por uma requisição da rede

Obs.: De forma especial, teremos os comentários padrões no JOB e em cada STEP do JOB:
Por exemplo, para JOB :

```
000001 //SSJ999XT JOB F02000,'PRODUCAO',CLASS=C,TIME=(1440,00),
000002 //              MSGCLASS=X,MSGLEVEL=(2,0)
000003 //*************************************************************
000004 //* DJ – (Descrição do Job)
000005 //*
000006 //* CP – (Condições de Processamento) Executa diariamente
```

```
000007 //*
000008 //* CR – (Condições de Reprocessamento) Voltar backup a partir ....
000009 //*
000010 //* PA – (PArticularidades) Executa após processamento do job
000011//*....
000012 //* RC – (Requisitos Críticos) Job que trata requisitos criticos.
000013 //*************************************************************
```

Por exemplo, para STEP :

```
000012 //JOBLIB   DD DSN=PD02.@00A01.LINKBAT,DISP=SHR
000013 //SSP001A1     EXEC PGM=ZP0PTH00,REGION=1024K
000014 //*************************************************************
000015 //* DS – (Descrição do Step)
000016 //*
000017 //* CP – (Condições de Processamento)
000018 //*
000019 //* CR – (Condições de Reprocessamento) Pode
reprocessar.
000020 //*
000021 //* PA – (PArticularidades) Precisa que o Job ....
000022 //*************************************************************
000023 //* RC – (Processa Requisito critico Sincronização de
Senha)..

000031 //SYSOUT    DD SYSOUT=X
000032 //SYSPRINT DD SYSOUT=X
```

STEP (de PROC's e JCL's)

Conjunto de comandos JCL que fazem parte de um JCL, de uma rotina ou de uma PROC. A regra que definirá a formatação para os nomes das PROC's será:

Sistema		Objeto	Seqüencial				Sufixo
1	2	3	4	5	6	7	8

SISTEMA: Identificador do Sistema, 2 caracteres alfabéticos.

OBJETO: Fixo para este objeto sistêmico em **S**.

SEQUENCIAL*: Seqüencial Alfa-Numérico.*

SUFIXO : Origem do step .

P : Programa

S : Sort
B : Backup
R : Recuperação

I : qualquer comando IBM.
C: requisito crítico

PROGRAMA ON LINE (TRANSACIONAL)

A regra que definirá a formatação para os nomes dos PROGRAMAS ON LINE será:

Sistema		Objeto	Identificador da Transação		Versão	Função	
1	2	3	4	5	6	7	8

SISTEMA: Identificador do Sistema, 2 caracteres alfabéticos.

OBJETO: Fixo para este objeto sistêmico em I.

IDENTIFICADOR DA TRANSAÇÃO: Seqüencial Alfa-Numérico.

VERSÃO: Número da versão do Programa. EM PRODUÇÃO SERÁ 0 (zero), em desenvolvimento poderá variar.

FUNÇÃO: Dois dígitos numéricos que respeitará as opções:
20 : Visualização, validação;
40 : Criação;
60 : Manutenção;
80 : Exclusão.

PROGRAMA ON LINE (CONVERSACIONAL) 3270

A regra que definirá a formatação para os nomes dos PROGRAMAS ON LINE (3270) será:

Sistema		Objeto	Seqüencial				
1	2	3	4	5	6	7	8

SISTEMA: Identificador do Sistema, 2 caracteres alfabéticos.

OBJETO: Fixo para este objeto sistêmico em I.

SEQUENCIAL: Seqüencial Alfa-Numérico.

PROGRAMA BATCH, DE ENTRADA DE DADOS E SHELL

A regra que definirá a formatação para os nomes dos PROGRAMAS BATCH, DE ENTRADA DE DADOS E SHELL será:

Sistema	Objet	Seqüencial

		o					
1	2	3	4	5	6	7	8

SISTEMA: Identificador do Sistema, 2 caracteres alfabéticos.

OBJETO: Fixo para este objeto sistêmico em B.

SEQUENCIAL: Seqüencial Alfa-Numérico.

TELAS, MAPSET CICS

A regra que definirá a formatação para os nomes das TELAS, MAPSETS CICS será:

Sistema		Objeto	Seqüencial				
1	2	3	4	5	6	7	8

SISTEMA: Identificador do Sistema, 2 caracteres alfabéticos.

OBJETO: Fixo para este objeto sistêmico em M.

SEQUENCIAL: Seqüencial Alfa-Numérico.

PACKAGE

Módulo que contém o caminho de acesso aos dados usados por uma aplicação;
A regra que definirá a formatação para os nomes dos PACKAGE's será:

Sistema		Objeto	Seqüencial				
1	2	3	4	5	6	7	8

SISTEMA: Identificador do Sistema, 2 caracteres alfabéticos.

OBJETO: Fixo para este objeto sistêmico em K.

SEQUENCIAL: Seqüencial Alfa-Numérico.

PLAN

Lista de Package;
A regra que definirá a formatação para os nomes dos PACKAGE's será:

Sistema		Objeto	Seqüencial				
1	2	3	4	5	6	7	8

SISTEMA: Identificador do Sistema, 2 caracteres alfabéticos.

OBJETO: Fixo para este objeto sistêmico em N.

SEQUENCIAL: Seqüencial Alfa-Numérico.

PROC

Conjunto de comandos JCL que serão submetidos ao Sistema Operacional MVS;
A regra que definirá a formatação para os nomes das PROC's será:

Sistema		Objeto	Seqüencial			
1	2	3	4	5	6	7

SISTEMA: Identificador do Sistema, 2 caracteres alfabéticos.

OBJETO: Fixo para este objeto sistêmico em H.

SEQUENCIAL: Seqüencial Alfa-Numérico.

COPY BOOK

Estrutura de dados que será utilizada por outros objetos sistêmicos;
A regra que definirá a formatação para os nomes dos COPY BOOK's será:

Sistema		Objeto	Tipo de Book	Seqüencial			
1	2	3	4	5	6	7	8

SISTEMA: Identificador do Sistema, 2 caracteres alfabéticos.

OBJETO: Fixo para este objeto sistêmico em C.

TIPO DE BOOK: Caractere que identificará o tipo de Copy Book.
D : DCLGEN
A : Array
K : Constante
L : Linkage
M : Mapas
P : Procedure
R : Relatório
V : MSG I/O
W : Working
E : Easytrieve
B : Serviço de dados batch
I : Serviço de dados on-line

SEQUENCIAL: Seqüencial Alfa-Numérico.

SUBROTINAS(ROTINA), FUNÇÃO

Subrotina: Código de uso geral que contenha comando de acesso a dados. Não existe a necessidade de retorno de um valor resultante do processo;
Função: Código de uso geral que não contenha comandos de acesso a dados. Deve haver o retorno de um valor resultante da manipulação dos parâmetros;
A regra que definirá a formatação para os nomes das SUBROTINAS(ROTINAS), FUNÇÃO será:

Sistema		Objeto	Identificador de Programa	Seqüencial			
1	2	3	4	5	6	7	8

SISTEMA: Identificador do Sistema, 2 caracteres alfabéticos.

OBJETO: Fixo para este objeto sistêmico em R.

IDENTIFICADOR DE PROGRAMA: Caractere que identificará o tipo de programa.
B : Batch
I : On Line

SEQUENCIAL: Seqüencial Alfa-Numérico.
para Serviços de Dados:
5 e 6: Cód. da tabela
7: Tipo de ação (4=Inserção ; 6=Alteração ; 8=Exclusão)
8 : Sequencial livre, pois pode existir mais de um serviço de dados para cada tipo de ação

para Rotinas Gerais (não associadas a nenhum serviço de dados):
5 e 6: Fixo "00"
7 e 8 : Sequencial livre

TRANSAÇÃO CICS

Transação CICS: Conjunto de um ou mais programas que executam uma função específica.
A regra que definirá a formatação para os nomes das TRANSAÇÕES CICIS será:

Sistema		Seqüencial	
1	2	3	4

SISTEMA: Identificador do Sistema, 2 caracteres alfabéticos.

SEQUENCIAL: Seqüencial Alfa-Numérico.

MENSAGENS

Mensagens: As mensagens para o sistema necessitam ser OBRIGATORIAMENTE cadastradas e caracterizadas no back-end (framework) como (I)nformativa, (E)rro Funcional ou (A)bend, para poderem ser usadas e a morfologia para seu cadastramento e codificação será:

Sistema		Seqüencial				
1	2	3	4	5	6	7

SISTEMA: Identificador do Sistema, 2 caracteres alfabéticos.

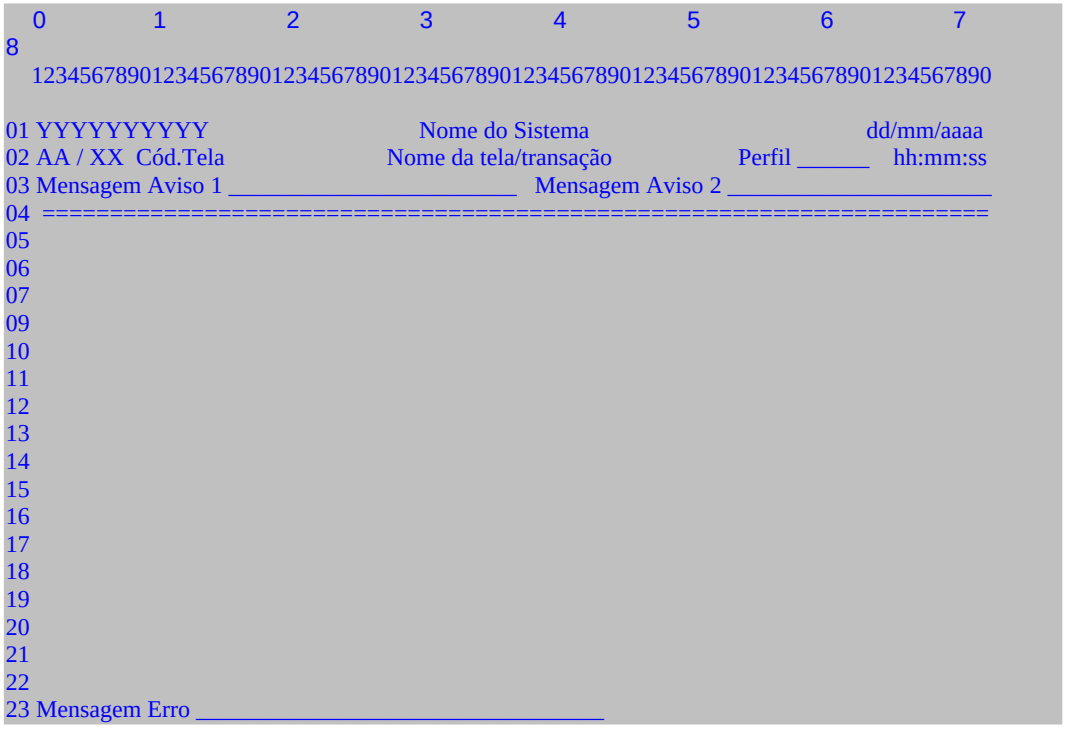
SEQUENCIAL: Seqüencial Numérico.

INTERFACE PADRÃO: TELA

PADRÃO TELA PF'S :
F1-Help = Help (Quando Abreviado : HP)
F2-FTP = Listar (Quando abreviado : FTP)
Transações de Manutenção em Detalhe – Ativar a transação de consulta em lista correlacionada à transação de manutenção;
Transações de Consulta em Lista – Ativar a transação de manutenção em detalhe correlacionada à transação de consulta em lista (selecionando uma linha da listagem e pressionando F2, é ativada a transação para manutenção dos dados presentes na linha selecionada).
F3-END = Retornar ou Sair (Quando abreviado : End)
F4-INS = Inserir (Quando abreviado : Ins)
F5-MOD = Alterar (Quando abreviado : Mod)
F6-DEL = Eliminar (Quando abreviado : Del)
F8-PG+ = Próxima Página (Quando abreviado : PG+)
F9-PG1 = Ir Primeira Página
ENTER = Consultar (Quando abreviado : Exec.)
CLEAR = Fim (Não pode ser abreviado).

ORIENTAÇÕES :
Campos com brilho : Mensagens e campos com erros;
YYYYYYYYY – Nome da empresa;
AA / XX – código da Transação, onde AA é o código do sistema e XX o seqüencial da transação;
Perfil – é o perfil do usuário;
Cód.Tela conterá código da tela (Mapa);
Data e Hora são obrigatórios;
A linha 23 é reservada como linha de mensagem;
A linha 24 é reservada como linha de PF's. A linha de PF's será preenchida segundo o padrão, as PF's vem primeiro, em ordem crescente, depois ENTER.
As mensagens devem ser claras e objetivas, orientando o usuário;

Para campos de taxas e percentuais, utilizar, na tela, o símbolo apropriado (%);
Para campos de indicadores direcionar a entrada de dados, por exemplo com (S/N);
Sempre que possível utilizar nas datas ANO com quatro posições;



NOMENCLATURA PARA ELEMENTOS DE PROGRAMAÇÃO

Verificar se já existe nomeclatura e adaptá-la para sistemas críticos.

PLATAFORMA BAIXA

PROGRAMAS VISUAL BASIC X.X

Nomenclatura para programas construídos em VB x.x.

Tipo de Programas	**Sistema**		**Objeto**	**Seqüencial**				**Sufixo**	**Hífen (Fixo)**	**Descritivo (Texto Livre 25 posições)**	**Extensão**
Posição	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10 a 25**	
MDI (form)	AA		F	0000 (Fixo)				Z	-	(texto livre)	FRM
Formulários	AA		F	9999				Z	-	(texto livre)	FRM
Módulos	AA		M	9999				Z	-	(texto livre)	BAS
Classes	AA		C	9999				Z	-	(texto livre)	CLS
(*) Relatórios	AA		L	9999				9	-	(texto livre)	RPT ou RDL

PROGRAMAS DOT.NET / ASP.NET

Nomenclatura para programas construídos em DOT.NET e ASP.NET.

Tipo de Programas	**Sistema**		**Objeto**	**Código Alfanumérico**				**Sufixo**	**Hífen (Fixo)**	**Descritivo (Texto Livre 25 posições)**	**Extensão**
Posição	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10 a 25**	
Global asa	Global							Z	-		asax
Config Files	Web							Z	-		config
Style Sheets	AA		Y	AAAA				Z	-	(texto livre)	CSS
Classes	AA		C	AAAA				Z	-	(texto livre)	VB ou CS
Web Forms	AA		W	AAAA				Z	-	(texto livre)	ASPX (aspx.vb aspx.cs)
Páginas HTML	AA		L	AAAA				Z	-	(texto livre)	HTM ou HTML
Windows Forms	AA		F	AAAA				Z	-	(texto livre)	VB ou CS
Web Services	AA		W	AAAA				Z	-	(texto livre)	ASMX (asmx.vb ou asmx.cs)
XSL Files	AA		Q	AAAA				Z	-	(texto livre)	XSL
XML Files	AA		X	AAAA				Z	-	(texto livre)	XML
(*) Relatórios	AA		L	AAAA				A	-	(texto livre)	RPT ou RDL

SISTEMA: Identificador do Sistema, 2 caracteres alfabéticos.

CÓDIGO ALFANUMÉRICO: Código identificador do arquivo (pode ser utilizado como seqüencial).

SUFIXO: Fixo em Z, com exceção para Relatórios que comporá o Código Alfanumérico com 5 posições.

NOMENCLATURA PARA ELEMENTOS DE PROGRAMAÇÃO

Métodos: verbos no infinitivo com pascal case (ex: EfetuarTransacao);

Eventos: verbos no particípio com pascal case (ex: EncerradaSessão);
Propriedades ou Variáveis Públicas: substantivos com pascal case (ex: ValorGarantia);
Variáveis Locais (variáveis de métodos): substantivos com camel case (ex: codigoProcesso);

Variáveis Privadas (variáveis de classe): substantivos com camel case iniciadas com "_" (ex: _valorMaximo);
Parâmetros: substantivos com camel case finalizados com "_" (ex: valorMinimo_);
Controles: substantivos prefixados com o tipo do controle com camel case (ex: txtNomeProcesso);
Constantes: substantivos com upper case, com o prefixo "C_" e separando cada palavra com "_" (ex: C_TAMANHO_REGISTRO).

IMPORTANTE: Se houver alguma dúvida ou este padrão não lhe atender em algum caso particular, procure o analista de qualidade.

ANEXO 4 QUESTIONÁRIO DE SATISFAÇÃO SOFTWARE DO ESTUDO DE CASO

DADOS DO AVALIADOR

Nome do Avaliador:
Cargo / Função:

LEGENDA

Performance					Grau de Importância				
MB Muito Bom	B Bom	RE Regular	RU Ruim	MR Muito Ruim	MG Muito Grande	G Grande	M Média	P Pouca	N Nenhuma

CARACTERÍSTICAS DE QUALIDADE

PERFORMANCE					CRITÉRIOS E SUBCRITÉRIOS DE QUALIDADE	IMPORTÂNCIA				
MB	B	RE	RU	MR		MG	G	M	P	N
FUNCIONALIDADE										
					– Adequação - Completitude - As funções do software especificadas na documentação foram todas implementadas; - As funções implementadas atendem de forma completa os objetivos declarados na documentação; - As funções implementadas satisfazem à necessidade da tarefa a que o produto se propõe realizar.					
					– Exatidão - Acurácia As funções verificadas no software geram resultados corretos ou conforme o esperado.					
					– Interoperabilidade - Interfaces O software interage corretamente com outros sistemas conforme especificado.					
					– Segurança de Acesso – Acesso Seletivo O software: - possibilita acesso seletivo, ou seja, usuários têm acesso a determinadas tarefas através de senhas; - é compatível com o tipo de informação que manipula; - impede a utilização das funções não					

PERFORMANCE					CRITÉRIOS E SUBCRITÉRIOS DE QUALIDADE	IMPORTÂNCIA				
MB	B	RE	RU	MR		MG	G	M	P	N
					autorizadas; - permite gerenciamento das senhas de acesso.					
CONFIABILIDADE										
					2.1 – Maturidade O software ou suas interfaces apresentam falhas com freqüência mínima ou quando apresentam essas falhas independem do software.					
					2.2 – Tolerância às falhas O software ou suas interfaces, quando apresentam falhas mantêm um nível de desempenho apropriado sem gerar danos indesejáveis.					
					2.3 – Recuperabilidade O software ou suas interfaces, quando apresentam falhas: - permitem que suas funcionalidades sejam restabelecidas rapidamente; - permitem que dados afetados sejam recuperados.					
USABILIDADE										
					3.1 – Inteligibilidade O usuário: - entende facilmente os conceitos lógicos utilizados no sistema - entende facilmente a aplicação.					
					3.2 – Aprendizibilidade O usuário aprendeu a usar sistema com facilidade.					
					3.3 – Operacionalidade O software é fácil de usar e possibilita um controle das operações realizadas.					
					3.4 – Atratividade O software é atrativo ao usuário e possui documentação e material de treinamento suficiente.					
EFICIÊNCIA										
					4.1 – Tempo de resposta As funções do software possuem tempo de processamento e resposta adequados.					

PERFORMANCE					CRITÉRIOS E SUBCRITÉRIOS DE QUALIDADE	IMPORTÂNCIA				
MB	B	RE	RU	MR		MG	G	M	P	N
					4.2 – Recursos Empregados O software evidencia adequadamente a quantidade de cada recurso utilizado e a duração da utilização (exemplos: memória, disco rígido, processador, requisições de banco de dados, rede, entre outros).					
MANUTENIBILIDADE										
					5.1 – Analisabilidade O software possibilita diagnosticar facilmente e rapidamente as causas de falhas e partes a serem modificadas, ou seja, é fácil encontrar uma falha quando ocorre.					
					5.2 – Modificabilidade O software possibilita facilmente modificar suas funcionalidades, remover defeitos ou adapta-lo a mudanças.					
					5.3 – Estabilidade O software frequentemente apresenta falhas após alterações/manutenções.					
					5.4 – Testabilidade O software é fácil de testar, ou seja, é fácil validar os requisitos implementados ou as manutenções realizadas.					
PORTABILIDADE										
					6.1 – Adaptabilidade O software é fácil de ser adaptado a outros ambientes.					
					6.2 – Capacidade para ser instalado O software é fácil de ser instalado tanto em seu ambiente padrão quanto em outros ambientes.					
					6.3 – Capacidade para substituir O software é fácil de ser substituído por outro software.					
					6.4 – Capacidade para co-existir O software pode co-existir com outros softwares em ambiente compartilhado.					
					Conformidade O software está de acordo com as normas, convenções ou regulamentos previstos em lei.					

LEGENDA AUXILIAR

RESUMO DOS CRITÉRIOS E SUBCRITÉRIOS DE QUALIDADE – ISO/IEC 9126

1 – FUNCIONALIDADE:

Definição: Conjunto de atributos do software que evidenciam a existência de um grupo de funções e suas propriedades especificadas. As funções são as que satisfazem as necessidades implícitas e explícitas.
Pergunta(s): o conjunto de funções satisfaz as necessidades implícitas e explícitas para a funcionalidade a que se destina o produto? Atende e/ou satisfaz as necessidades explícitas e implícitas? Observações: a avaliação dessa característica é, basicamente, uma comparação entre o grupo ideal de funções para o objetivo a que se propõe o produto de software e o que este apresenta. É a principal característica de qualidade para qualquer tipo de software. Em outras palavras, mede as capacidades, isto é, o conjunto de funções oferecidas aos usuários que satisfaçam suas necessidades.

1.1 – Adequação
Definição: Atributos do software que evidenciam a presença de um conjunto de funções e sua apropriação para as tarefas especificadas.
Pergunta(s): Propõe-se a fazer o que é apropriado?

1.2 – Exatidão
Definição: Atributos do software que evidenciam a geração de resultado ou efeitos corretos ou conforme acordados.
Pergunta(s): Faz o que foi proposto de forma correta? Gera resultados corretos conforme acordados?

1.3 – Interoperabilidade
Definição: Atributos do software que evidenciam sua capacidade de interagir com sistemas especificados.
Pergunta(s): É capaz de interagir com os sistemas especificados?

1.4 – Segurança de Acesso
Definição: Atributos do software que evidenciam sua capacidade de evitar o acesso não autorizado, acidental ou deliberado, a programas e dados.
Pergunta(s): Evita acesso não autorizado, acidental ou deliberado, a programas e dados?

1.5 – Conformidade
Definição: Atributos do software que o possibilitam estar adequado às normas, convenções ou regulamentações previstas em lei e descrições similares, relacionadas à aplicação.Pergunta(s): Está de acordo com as normas, convenções, regulamentações e descrições similares?

1 – CONFIABILIDADE

Definição: Conjunto de atributos que evidenciam a capacidade do software de manter seu nível de desempenho sob condições estabelecidas durante um período de tempo estabelecido.
Pergunta(s): é imune a falhas? O desempenho se mantém ao longo do tempo e em condições estabelecidas? Observação: esta característica refere-se à capacidade do software funcionar de acordo com a sua especificação funcional, sendo determinada por fatores como segurança, ausência de falhas e resultados corretos.

2.1 – Maturidade
Definição: Atributos do software que evidenciam a frequência de falhas por defeitos no software.
Pergunta(s): Com que frequência apresenta falhas?

2.2 – Tolerância a Falhas
Definição: Atributos do software que evidenciam sua capacidade de manter um nível de desempenho especificado nos casos de falhas no software ou de violação nas interfaces especificadas.
Pergunta(s): Ocorrendo falhas como ele reage?

2.3 – Recuperabilidade
Definição: Atributos do software que evidenciam sua capacidade de restabelecer seu nível de desempenho e recuperar os dados diretamente afetados, em caso de falha, e o tempo e esforço necessário para tal.
Pergunta(s): É capaz de recuperar dados em caso de falha?

2.4 – Conformidade
Definição: Atributos do software que o tornam consoantes com padrões ou convenções relacionados à confiabilidade.
Pergunta(s): Está de acordo com padrões ou convenções de confiabilidade?

3 – USABILIDADE

Definição: Conjunto de atributos que evidenciam o esforço necessário para se poder utilizar o software, bem como o julgamento individual desse uso, por um conjunto explícito ou implícito de usuários.
Pergunta(s): É fácil de usar / utilizar? Observações: esta característica trata-se de um conjunto de atributos de software relacionado ao esforço necessário para o seu uso por determinado conjunto de usuários. Isto é, avalia o esforço necessário ao uso do software e ao seu aprendizado (treinamento). É um critério normalmente determinado por fatores como: interface, easy-to-use, documentos e material de treinamento adequado.

3.1 – Inteligibilidade
Definição: Atributos do software que evidenciam o esforço do usuário para reconhecer o conceito lógico e sua aplicabilidade.
Pergunta(s): É fácil entender os conceitos utilizados e a aplicação?

3.2 – Aprendizibilidade
Definição: Atributos do software que evidenciam o esforço do usuário para aprender sua aplicação.

Pergunta(s): É fácil aprender a usar / utilizar? Observações: Exemplos: controle de operação, entradas, saídas.

3.3 – Operabilidade
Definição: Atributos do software que evidenciam o esforço do usuário para sua operação e controle de sua operação.
Pergunta(s): É fácil de operar e controlar a operação?

3.4 – Atratividade
Definição: Atributos do software que evidenciam sua capacidade de atrair ao usuário. Pergunta(s): É atrativo ao usuário?

3.5 – Conformidade
Definição: Atributos do software que o tornam consoantes com padrões ou convenções relacionados à usabilidade.
Pergunta(s): Está de acordo com padrões ou convenções de usabilidade?

4 – EFICIÊNCIA

Definição: Conjunto de atributos que evidenciam o relacionamento entre o nível de desempenho do software e a quantidade de recursos usados, sob condições estabelecidas.
Pergunta(s): Os recursos e os tempos utilizados são compatíveis com o nível de desempenho requerido para o produto? É rápido e “enxuto”? Observação: esta característica se refere ao desempenho, tempo de resposta e uso eficaz dos recursos do sistema, sendo medido pela relação: nível de desempenho do software X quantidade de recursos utilizados.

4.1 – Tempo de Resposta
Definição: Atributos do software que evidenciam seu tempo de resposta, tempo de processamento e velocidade de execução de suas funções.
Pergunta(s): Qual é o tempo de resposta e de processamento e a velocidade de execução?

4.2 – Recursos Empregados
Definição: Atributos do software que evidenciam a quantidade de recursos usados e a duração de seu uso na execução de suas tarefas.
Pergunta(s): Quanto recurso utiliza? Qual recurso utiliza? Durante quanto tempo?

4.3 – Conformidade
Definição: Atributos do software que o tornam consoantes com padrões ou convenções relacionados à eficiência.
Pergunta(s): Está de acordo com padrões ou convenções de eficiência?

5 – MANUTENIBILIDADE

Definição: Conjunto de atributos que evidenciam o esforço necessário para fazer modificações especificadas no software. As modificações podem incluir correções,

melhorias ou adaptações no software devido a mudanças no ambiente ou nos seus requisitos.
Pergunta(s): É fácil de alterar e de modificar? Há facilidade para correções, atualizações e alterações? Observações: a avaliação pode ser feita através da análise da facilidade / rapidez de manutenção, custo das manutenções, tempo médio entre falhas (MTBF – Medium Time Between Failures), tempo médio de reparo (MTTR – Medium Time To Repair).

5.1 – Analisabilidade
Definição: Atributos do software que evidenciam o esforço necessário para diagnosticar deficiências ou causas de falhas, ou para identificar partes a serem modificadas.
Pergunta(s): É fácil encontrar uma falha quando ocorre?

5.2 – Modificabilidade
Definição: Atributos do software que evidenciam o esforço necessário para modificá-lo, remover seus defeitos ou adaptá-lo a mudanças ambientais.
Pergunta(s): É fácil modificar, adaptar e remover defeitos?

5.3 – Estabilidade
Definição: Atributos do software que evidenciam o risco de efeitos inesperados ocasionados por modificações.
Pergunta(s): Há grandes riscos de bugs quando se faz alterações?

5.4 – Testabilidade
Definição: Atributos do software que evidenciam o esforço necessário para validar o software modificado.
Pergunta(s): É fácil testar quando se faz alterações?

5.5 – Conformidade
Definição: Atributos do software que o tornam consoantes com padrões ou convenções relacionados à manutenibilidade.
Pergunta(s): Está de acordo com padrões ou convenções de manutenibilidade?

6 – PORTABILIDADE

Definição: Conjunto de atributos que evidenciam a capacidade do software de ser transferido de um ambiente para outro.
Pergunta(s): É fácil de usar / utilizar em outro ambiente? É possível utilizar o produto em diversas plataformas com pequeno esforço de adaptação? Observações: esta característica avalia basicamente o esforço necessário para que se faça transferências de ambientes do software para diferentes sistemas ou plataformas.

6.1 – Adaptabilidade
Definição: Atributos do software que evidenciam sua capacidade de ser adaptado a ambientes diferentes especificados, sem a necessidade de aplicação de outras ações ou meios além daqueles fornecidos para esta finalidade pelo software considerado.

Pergunta(s): É fácil adaptar a outros ambientes sem aplicar outras ações ou meios além dos fornecidos para esta funcionalidade no software considerado? É fácil adaptar a outros ambientes?

6.2 – Capacidade para ser Instalado
Definição: Atributos do software que evidenciam o esforço necessário para sua instalação num ambiente especificado.
Pergunta(s): É fácil instalar em outros ambientes?

6.3 – Capacidade para Substituir
Definição: Atributos do software que evidenciam sua capacidade e esforço necessário para substituir um outro software, no ambiente estabelecido para esse outro software.
Pergunta(s): É fácil substituir por outro software? Ë fácil usar para substituir outro software?

6.4 – Capacidade de Coexistir
Definição: Atributos do software que evidenciam sua capacidade de co-existir com outros softwares em ambiente compartilhado.
Pergunta(s): É possível coexistir com outros softwares em ambiente compartilhado?

6.5 – Conformidade
Definição: Atributos do software que o tornam consoantes com padrões ou convenções relacionados à portabilidade.
Pergunta(s): Está de acordo com padrões ou convenções de portabilidade?

ANEXO 5 FRAMEWORK DE PLANO DE TESTE

PLANO DE TESTE
<NOME DO PROJETO>

[O Plano de Testes apresenta o planejamento dos testes a serem realizados, incluindo detalhamento dos estágios e tipos de testes previstos para garantir a conformidade do produto com os requisitos identificados e, conseqüentemente, a aceitação do cliente. Essa sessão pode ser excluída após o preenchimento do documento.]

1. Objetivo

[Esta seção define o propósito deste documento].

Sobre o Produto

[Esta seção descreve resumidamente as principais características do aplicativo ou módulo a ser testado].

1.1 Escopo do Projeto de Teste

[Descreve a cobertura dos testes em relação ao produto, definindo o que vai ser testado num nível macro. Ex: Módulos a serem testados: (Marcação de Consulta, Cadastro de Paciente)].
Fora do Escopo do Projeto de Teste
[Esta seção é opcional. Relaciona os itens não cobertos pelo plano, de forma a garantir que todos os itens foram considerados].

2. Alvos de Teste

2.1 Testes de Casos de Uso

[Esta seção deve identificar os Casos de Uso a serem testados, a partir dos artefatos de especificação de requisitos (Documento de Requisitos, Documento de Visão, Especificação Suplementar, Estudo do Modelo de Casos de Uso)].

ID	Caso de uso	Observações
[Identificador] (Ex.: UCXX)	[Ex.: Caso de uso Cadastrar Cliente]	[Indicar a prioridade, importância ou percentual de ocorrência do caso de uso para o modulo a ser testado de acordo com o tipo de teste, ou quaisquer outras observações necessárias].

3. Critérios de Conclusão e Êxito

[Esta seção contém as condições que devem ser satisfeitas e estado que deve ser atingido para que o conjunto de testes seja considerado bem sucedido. Podem tomar como base: Critérios de cobertura dos testes (todas as funções foram exercitadas e as saídas observadas de acordo com o esperado), objetivos de qualidade atingidos].

4. Recursos Necessários

4.1Ambiente de Teste

[Os quadros abaixo apresentam sugestões para indicação dos recursos de hardware e software necessários para a execução dos testes].

Recursos de Hardware			
[Equipamento]	**[CPU]**	**[Memória (Mby)]**	**[Disco (Gby)]**

Recursos de Software	
[Descrição]	**[Versão]**

4.1.1Recursos Humanos

[O quadro abaixo apresenta os perfis e os profissionais designados no contexto das atividades relativas aos testes].

Recursos Humanos	
Perfil	**Responsável**
[Gerente de Teste]	
[Gestor de Teste]	
[Projetista de Teste]	
[Testador]	

5. Produtos Gerados

[Esta seção relaciona os artefatos que serão produzidos durante o processo de teste. Ex: (Casos de Teste, Procedimentos de Teste, Script de Teste, Resultado dos Testes)].

6. Cronograma de Atividades

[Esta seção deve conter uma lista detalhada de tarefas cobertas pelo plano, em ordem de dependência entre elas, com data de início e término previstos, e responsável conforme exemplo, abaixo],

Atividades	Responsável	Data Prevista de Inicio	Data Prevista Fim
Elaborar Registro de Ocorrência do Projeto	Gerente de Teste		
Elaborar Plano de Testes	Gestor de Teste		
Iniciar preenchimento do Relatório Progresso do Projeto (*)	Gestor de Teste		
Iniciar Métricas do Projeto de Teste (**)	Gestor de Teste		
Elaborar Casos de Teste	Projetista de Teste		

Elaborar Procedimentos de Teste	Projetista de Teste		
Preparar Ambiente de Teste	Projetista de Teste		
Executar Testes	Testador		
Concluir Casos de Teste ou Procedimentos de Teste (***).	Testador		
Registrar Falhas	Testador		
Consolidar Resultados do Teste	Testador		
Avaliar Resultados dos Testes	Gestor de Teste		
Elaborar o Relatório de Resultado dos Testes	Gestor de Teste		
Promover Reunião de Encerramento	Gerente de Teste		
Encaminhar Resultados do Teste	Gerente de Teste		
Concluir Métricas do Projeto de Teste	Gerente de Teste		
Concluir Relatório de Progresso de Projeto	Gerente de Teste		

(*) O Relatório do Progresso do Projeto deve ser atualizado através das reuniões de acompanhamento ao final de cada atividade do projeto de teste.

(**) A elaboração das Métricas do Projeto de Teste é executada em duas fases, Na primeira fase são registrados a “Meta” e os “Valores Aceitáveis” do projeto de teste e na segunda fase são registrados os valores “Apurados” e o “Desvio” obtido para cada métrica do projeto de teste.

(***) A coluna “Resultado Obtido” do formulário Casos de Teste só se aplica para Testes de Instalação. Nos demais tipos de teste esta coluna deve ser preenchida no formulário Procedimentos de Teste.

7. Riscos e Contingências

[Esta seção define os riscos e contingências envolvidos nos testes do plano (Ex.: Se o usuário não estiver disponível para o teste, um segundo usuário deve ser convocado)].

8. Referências

[Lista todos os documentos que foram utilizados como referência para o desenvolvimento deste documento, inclusive artefatos do PDS que venham a ser utilizados (Ex.: Documento de Visão, Especificação Suplementar e outros)].

Documento	Data de Criação (*)	Fonte de Origem

(*) Também pode ser indicada a data da consulta ao documento publicado na Internet

ANEXO 6 VISÃO GERAL DO SISTEMA PILOTO

Projeto Nome	Controle de Garantias			**Projeto Código**	
Projeto Gerente				**SSI Número**	
Elaborado por		**Em**			
Revisado por		**Em**		**Versão**	**1.0**

1. Descrição Geral

1.1 Objetivo

O novo Controle de Garantias deverá ser eficiente e flexível para gerenciar, de forma integrada e abrangente, todas as garantias oferecidas pelos clientes do Sistema Financeiro ABC em operações de crédito.
Ele operará em escala e administrará vários tipos de garantia: gerais(contrato e/ou nota promissória sem aval) e especiais, pessoais(aval, fiança, seguro ou caução) e reais(hipotecária, alienação fiduciária, penhor financeiro, penhor mercantil, penhor de direito ou consignação de rendimentos), compromissos(consignação de receitas, carta compromisso, hipoteca, cheques pré-datados ou duplicatas).

1.2 Escopo

- Análise da Proposta de Crédito
- Consolidar a política de exigência de garantias para suporte de operações de crédito (Garantia vs. Produto de Crédito);
- Proporcionar a avaliação de risco do cliente e da operação;
- Padronizar os procedimentos de análise, vinculação e cadastramento de garantias;
- Validar se as garantias oferecidas pelo cliente podem, efetivamente, suportar a operação de crédito proposta (em termos de valor, prazo e natureza);
- Disponibilizar responsabilidades efetivas (operações de crédito contratadas) e potenciais (avais prestados) dos clientes do Banco;
- Permitir a utilização correta de garantias dinâmicas (p.ex:fundos, títulos).
-
- Contratação da Operação
- Validar para que a vinculação da garantia exigida para um determinado produto seja efetivamente realizada antes da contratação da operação;
- Contribuir para a correta caracterização das garantias, as quais suportam as operações de crédito (em especial as garantias reais);
- Conhecer a localização física das garantias bem como controlar o seu processo de constituição;
- Permitir, se aplicável, que as garantias cadastradas e não utilizadas suportem novas operações de crédito;
- Suportar operações em moeda estrangeira.
-
- Administração das Garantias
- Registrar automaticamente os movimentos de utilização e liberação do saldo da garantia (este aspecto é particularmente importante quando se quer utilizar uma garantia já existente para novos créditos);
- Controlar o valor e o vencimento da garantia (alguns tipos de garantia podem estar sujeitas a processos de reavaliação periódica);
- Facilitar a negociação de alterações contratuais (p.ex:alteração do prazo ou montante da operação de crédito) e amarrar as condições da operação a uma nova estrutura de garantias compatível com a política do BANCO ABC.
-
- Gestão da Inadimplência
- Assegurar o rigor no cálculo das provisões, através do controle e da administração das garantias, vinculando estas com os respectivos códigos BACEN;

- Facilitar os processos nas agências e no departamento jurídico, agilizando o processo de recuperação de créditos vencidos e não pagos.

1.3 Benefícios Esperados

- Integrado com todas as carteiras de administração de crédito;
- Abrangente, cobrindo todos os tipos de garantias e de crédito;
- Parametrizável, permitindo alteração dos critérios de negócios de forma eficiente e rápida;
- Atualizado, garantindo que a informação sobre a garantia reflita sempre a realidade do momento e cobrindo de forma eficaz as operações de crédito a que está associada;
- Caracterizador, permitindo descrever com rigor as garantias cadastradas.

1.4 Público Alvo

- Gerentes operacionais para proposição de operações de crédito e definição de garantias.
- Apoiar a área de análise de crédito para análise das garantias.
- Comitê de Crédito para aprovação das operações de crédito baseado nas garantias.
- Apoiar a área jurídica nos processos de recuperação de créditos.
- Auditoria Interna na verificação das operações de crédito e suas garantias.
- Outros sistemas que necessitam de informações das garantias do cliente.

1.5 Usuários do Sistema

Tipo de Usuário	Usuário	Caso
Crédito	Analista	Definir características de cada garantia
		Definir documentos necessários para cada garantia
		Definir relacionamento entre produto crédito e produto garantia
		Definir percentual de garantia exigido
		Cadastrar Regras de Distribuição Específicas na conta lastro
Operacional	Agência	Cadastrar Reciprocidade no Perfil da Lastro
		Cadastrar informações básicas da Conta Lastro
		Cadastrar informações complementares da Conta Lastro
		Cadastrar relacionamento Conta Lastro - Avalistas
		Cadastrar relacionamento Conta Lastro - Contrato Crédito
		Alterar dados cadastrais de garantias
		Manutenção do saldo disponível da Conta Lastro
		Registrar avaliações e vistorias
		Manter Lista Negra de Sacados
		Comandar Bloqueios de Garantias
Sistêmico	Controle Garantias	Calcular concentração de sacados nas contas vínculadas
		Disparar alerta para dados incompatíveis com a parametrização

		Proceder a checagens necessárias para implantação da operação
		Proceder a bloqueio das garantias nos sistemas produto
		Manutenção saldo disponível da Conta Lastro
		Disparar alerta para dados cadastrais de garantias incompatíveis
		Controlar datas para reavaliação de Garantias
		Detectar Insuficiência de Garantias
		Cancelar Conta Lastro
		Reavaliar taxa cambial
		Efetuar contabilização
		Registrar histórico das alterações
		Registrar erro de cadastramento/vinculação da garantia
Comitês	Validadores	Aprovar liberação operação sem documentação completa
Back Office	Usuário	Alterar agência em Conta Lastro
Corporativo	Corporativo	Efetuar consultas/requisitar relatórios disponíveis no Sistema de Controle de Garantias

2. Especificação do Negócio

2.1 Situação Atual / Solução Proposta

Situação Atual

O Sistema de Garantias tem o perfil de consolidador de informações. Não possibilita alterações de dados recebidos dos sistemas produtos (contas e valores), exceto para os dados de avalistas e alienados, cadastrados diretamente no Sistema de Garantias, e porcentagem de garantia. Seus principais usuários são as Agências, Regionais, Depto. de Análise de Crédito e Departamento de Ativos e Contenciosos.

O cadastro de porcentagem de garantia nas operações dos Sistemas Produtos é restrito a 1 byte. Essa informação é convertida, quando enviada para o Sistema de Garantias, no valor máximo do intervalo representado. O Sistema de Garantias permite, para efeito de enquadramento das garantias, alteração manual desse percentual.

O Sistema de Garantias apresenta deficiência na consolidação dos dados dos diversos sistemas. O caso crítico está relacionado ao uso do número de contrato divergente, o que pode gerar relacionamentos errados (contrato-garantia). Para o caso em que não seja encontrado o contrato relacionado a uma garantia, esta aparece no total de garantias do cliente, mas aparece sem contrato relacionado, necessitando de acerto manual posterior.

O Sistema de Garantias não possui bases históricas da relação garantias-contratos.
As reavaliações das garantias alienadas são controladas manualmente através de consultas no Sistema de Garantias.

O departamento de ativos criou, a partir de arquivo de enquadramento de garantias, funcionalidades para atender necessidades não atendidas pelo Sistema de Garantias.O Sistema de Garantias não valida as informações de tipo de garantias das operações de crédito com as garantias existentes, preocupando-se somente para que os valores de garantias exigidos estejam devidamente enquadrados.

Situação Proposta

O novo Controle de Garantias irá retratar em um único local as garantias de um cliente existentes nas operações de crédito, facilitando as análises e aprovações de crédito do mesmo. Através das pesquisas realizadas junto ao Cadastro de Clientes, Proposta, Garantias e Central de Risco, os responsáveis pela análise e aprovação de crédito poderão mensurar de uma forma mais segura e eficiente as operações de crédito.

Maiores detalhes a respeito da situação proposta podem ser vistos na apresentação PowerPoint na base de conhecimento do projeto.

2.2 Regras de Negócios

Entradas e Interfaces

- O sistema deverá alertar o usuário sobre a inclusão de garantias a vencer após o vencimento do contrato da operação;
- A lógica de numeração da Conta Lastro será definida pelo BANCO ABC. Sugere-se um número seqüencial ao da formalização da operação
- Os dados de clientes serão acessados através de consulta do Controle de Garantias ao novo Cadastro de Clientes;
- Os dados padrão(produtos, empréstimos, garantias e porcentagem de garantia) deverão ser apresentados em listas para cadastramento pelo usuário, através do Controle de Garantias e das Tabelas Corporativas;
- Após o cadastramento do contrato de operação relacionado à garantia, através de interface online dos sistemas legados, o Controle de Garantias validará a recepção dos documentos necessários para a garantia. Após a validação, deverá atualizar os sistemas produtos, conforme critérios de associação contrato-garantia obtidos no levantamento da situação atual;
- O Controle de Garantias irá prever a necessidade de aprovação facultativa, para os casos em que julgar de baixo risco a falta de algum documento, da garantia para liberação da operação. Deverá ser registrados o usuário aprovador, a data/hora da aprovação e a pendência de documentação;
- Para o sistema de Propostas, o Controle de Garantias será o provedor de informações do relacionamento operação de crédito-garantia e parâmetros associados a esse relacionamento (porcentagem de garantia, valores máximos e mínimos de crédito), bem como a inclusão da Conta Lastro da proposta em elaboração.

Armazenamento de dados

- Estarão armazenados no Controle de Garantias os dados de garantias alienadas. Para as demais garantias, estarão armazenados os relacionamentos Conta Lastro-conta contrato do produto em seu sistema;
- O Controle de Garantias deverá atualizar, automaticamente para as contas garantias, os valores de saldo disponível, valor global, data de criação, situação de conta e data de última situação, conforme dados recebidos diariamente dos sistemas produtos em processamento batch;
- Não será permitida a exclusão da Conta Lastro se ela estiver ou esteve vinculada a uma operação de crédito, ou quando possua dados complementares associados;
- Qualquer alteração de dados deverá sofrer validações em relação às regras de cadastramento (parametrização) e qualquer discrepância deverá ser alertada pelo Sistema;

- O sistema deverá estar preparado para contabilização dos eventos de vinculação e liberação da garantia, alterações de natureza jurídica do primeiro titular, bem como reavaliações cambiais, entretanto não estará integrado ao processo de contabilização atual do BANCO ABC neste primeiro momento. Esta funcionalidade será efetivada com a substituição dos demais sistemas legados;
- Periodicamente, o sistema deverá excluir os registros de Conta Lastro não associada a nenhuma operação de crédito em um período a ser determinado pelo BANCO ABC.

2.3 Restrições Gerais

As restrições de uso serão definidas pelo gestor do sistema de Garantias, permitindo acesso aos perfis específicos. Todo o controle de acesso será feito pela Arquitetura, permitindo ao usuário utilizar somente as funcionalidades que ele tem acesso.

2.4 Requisitos Técnicos e Legais

O Controle de Garantias será executado sobre a nova plataforma de processamento do Banco ABC. A especificação técnica dessa plataforma está na documentação do projeto de Arquitetura.
As Tabelas Corporativas e as tabelas do Cadastro de Clientes deverão estar disponíveis para acesso do Controle de Garantias.
Os serviços batch serão agendados para processamento noturno, basicamente gerando relatórios e recebendo dados das interfaces dos sistemas produtos. Estes serviços serão agendados na ferramenta adotada pelo Sistema Financeiro ABC, sendo executados segundo a ordem de prioridade e precedência definida. Os sistemas produtos também atualizarão a base de dados do sistema Controle de Garantias. Ainda não foi dimensionado o tamanho e o crescimento da base de dados, não sendo possível, neste momento, prever o tempo de execução dos serviços agendados.

2.5 Nível de Serviço

O Controle de Garantias estará disponível 24 horas por dia, durante 7 dias na semana. O tempo de resposta não poderá impactr a utilização pelo usuário.
Os critérios de expurgo dos dados serão definidos durante a revisão funcional com os usuários.

2.6 Indicadores

Previsão inicial dos indicadores:

- Tempo máximo de resposta para consultas cadastrais;
- Tempo máximo de resposta para consultas de históricos.

2.7 Volumes Esperados

Volumes de dados do sistema atual

Data base: novembro/2004

Quantidades em garantias:

Duplicatas	100000	11%
cheques	800000	87%
alienadas	9000	1%
aplicações	10000	1%
	919000	

Garantias em valores

duplicatas	766.931.000	22%
cheques	766.931.000	22%
alienadas	1.444.789.000	42%
aplicações	447.987.000	13%
	3.426.638.000	

3.2 Funcionalidades Macro com Descrição

- Parametrização: definição das políticas para cada tipo de garantia;
- Garantias: Criação das Características da Conta Lastro, Formalização de Garantiase Gestão de Garantias
- Consultas: consultas feitas on-line ou através de relatórios, onde o analisador ou aprovador poderá visualizar as garantias atuais e históricas do cliente;
- Dados Cadastrais: consulta aos dados do cliente existentes no Cadastro de Cliente e na Central de Risco, da proposta existente no sistema de Proposta.

4. Segurança / Auditagem

A Arquitetura é responsável pelo controle de segurança, e por gerar os logs para Auditoria. O sistema controla a confidencialidade dos dados, através da hierarquia da informação e o usuário logado somente poderá ver os dados que lhe são permitidos.