UNIVERSIDADE FEDERAL DO MARANHÃO

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

CURSO DE CIÊNCIA DA COMPUTAÇÃO

WELSSON HOLANDA OLIVEIRA

CONFIABILIDADE E PROTEÇÃO DE SOFTWARE –

Uma proposta para um processo confiável de software

São Luís

2013

WELSSON HOLANDA OLIVEIRA

CONFIABILIDADE E PROTEÇÃO DE SOFTWARE –

Uma proposta para um processo confiável de software

Monografia apresentada ao Curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Orientadora: Prof.ª Msc. Maria Auxiliadora Freire

São Luís

2013

Oliveira, Welsson Holanda.

Confiabilidade e proteção de software – uma proposta para um processo confiável de software/ Welsson Holanda Oliveira. – São Luís, 2013.

60 f.

Impresso por computador (fotocópia).

Orientadora: Maria Auxiliadora Freire

Monografia (Graduação) – Universidade Federal do Maranhão, Curso de Ciência da Computação, 2013.

1. Software – Confiabilidade. I Título.

CDU 004.052

WELSSON HOLANDA OLIVEIRA

CONFIABILIDADE E PROTEÇÃO DE SOFTWARE –

Uma proposta para um processo confiável de software

Monografia apresentada ao Curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Aprovada em: 07 / 03 / 2013

BANCA EXAMINADORA

Prof.ª Maria Auxiliadora Freire (Orientadora)
Mestre em Ciência de Engenharia
Universidade Federal do Maranhão

Prof. Carlos Eduardo Portela Serra de Castro
Mestre em Informática
Universidade Federal do Maranhão

Prof. Mário Antônio Meireles Teixeira
Doutor em Ciências de Computação
Universidade Federal do Maranhão

Aos meus pais,
Luciene Holanda Oliveira e
José Ezaquiel Maia de Oliveira

AGRADECIMENTOS

A Deus, pelas bênçãos que tem colocado em meu caminho e por ter-me proporcionado mais essa vitória.

Aos meus pais, pelo incentivo, apoio, dedicação, e por sempre estarem ao meu lado me dando o apoio e a força necessária para persistir nos meus objetivos.

As minhas irmãs, por sempre acreditarem no meu esforço, pelo incentivo dado e pela compreensão de nem sempre está presente.

A minha avó Valdericima, por ser um exemplo de vida e superação, e ser uma pessoa tão maravilhosa e adorável.

A todos meus tios e tias, primos e primas, que sempre torceram por mim e me deram muita força nessa longa jornada.

Aos meus queridos amigos e amigas, Darlene, Karyne, Naires, Marcos e Tiago, pelo incentivo, por estarem sempre ao meu lado durante todos esses anos, mesmo distantes, mas ao mesmo tempo presentes.

A Eloíde e o Milton, pela oportunidade que a mim foi concedida, por me acolherem em sua residência, e por acreditarem no meu potencial.

Aos meus colegas de curso, em especial, André, Alessandro, Fábio, Felipe, Genilson, Ivo, Keila, Júnior, Jeferson, Leonardo, Márcio, Pablo, Valéria, Wagner, pelo companheirismo, pela ajuda e esforço demandados durante todos esses anos.

A todos os professores, ao coordenador do curso e a professora Auxiliadora, pela orientação, pela paciência, os ensinamentos, e por sua dedicação durante a graduação e principalmente nessa etapa final.

A todas as pessoas que sempre torceram por mim e acreditaram na minha capacidade.

“Para realizar grandes conquistas,
devemos não apenas agir,
mas também sonhar;
não apenas planejar,
mas também acreditar.”

(Anatole France)

RESUMO

Uma das grandes preocupações dos engenheiros de software é produzir sistemas que possam ser confiáveis para seus usuários e que tenham a proteção necessária contra falhas. Por isso, é preciso que o processo de desenvolvimento de todo o projeto de software possua técnicas e métodos que possam proporcionar ao sistema a confiabilidade e a disponibilidade que dele se espera. Dessa forma, são empregadas recursos de autoverificação para descobrir os defeitos antes que eles ocasionem falha. É necessário fazer um gerenciamento de riscos de todo o projeto, para identificá-los e possam-se adotar meios de prevenção. Um estudo de confiabilidade deve ser feito para descobrir quais tipos de falha o sistema está propício, para que se tomem as medidas de evitar seus efeitos. Algumas métricas são utilizadas para medir a confiabilidade do software, como a probabilidade de falha sob demanda (POFOD), a taxa de ocorrência de falhas (ROCOF), e a medida de disponibilidade (AVAIL). A redundância e diversidade de componentes é uma das técnicas adotadas como meio de tolerar falhas do sistema. Também, são utilizadas arquiteturas tolerantes a defeitos como mecanismo de tolerância a falha. Após a conclusão do desenvolvimento do sistema é preciso fazer testes que possam medir a confiabilidade do software, como testes de tempo de ocorrência de falhas, intervalo entre falhas e quanto tempo leva para reparar as falhas. Todas as funcionalidades do sistema são testadas para garantir seu perfeito funcionamento. Além disso, são feitos testes para descobrir a porcentagem que o sistema está disponível para atender seus usuários. Ao final desse trabalho, é proposto um processo confiável de software, que vai garantir que toda a construção do sistema seja feita na mais absoluta cautela seguindo todas as especificações necessárias para obter a confiabilidade que se deseja alcançar para o sistema.

Palavras-chave: proteção, projeto de software, confiabilidade, disponibilidade, gerenciamento de riscos, *Probability of Failure on Demand* (POFOD), *Rate of Occurrence of Failures* (ROCOF), *Availability* (AVAIL), redundância, diversidade, tolerância a falha, processo confiável de software.

ABSTRACT

A major concern of software engineers is to produce systems that can be trusted for its users and they have the necessary protection against failures. Therefore it is necessary that the process of development of the whole software project has techniques and methods that can provide the system reliability and availability that is expected of it. Thus, self-checking features are used to find defects before them to bring on failure. You must make a risk management throughout the project, to identify them and can be adopted means of prevention. A reliability study should be done to find out which types of failures the system is conducive to those taking measures to avoid its effects. Some metrics are used to measure software reliability, as the probability of failure on demand (POFOD), the rate of occurrence of failures (ROCOF), and the extent of availability (AVAIL). The diversity and redundancy of components is a technique adopted as a means for tolerating faults in the system. Also, they are used as defect tolerant architectures fault tolerance mechanism. Upon completion of the system development is necessary to do tests that measure software reliability, as time tests of failures, time between failures and how long it takes to repair faults. All features of the system are tested to ensure flawless performance. In addition, tests are done to find out the percentage that the system is available to meet your users. At the end of this paper, we propose reliable process software, which will ensure that all construction system is made in absolute caution following all the specifications required to achieve the reliability you want to achieve for the system.

Keywords: protection, software design, reliability, availability, risk management, Probability of Failure on Demand (POFOD) Rate of Occurrence of Failures (ROCOF) Availability (AVAIL), redundancy, diversity, fault tolerance, process reliable software.

LISTA DE FIGURAS E ILUSTRAÇÃO

LISTA DE ABREVIATURAS E SIGLAS

ALARP	*As Low as Reasonably Practicable*
AVAIL	*Availability*
ECC	*Error-correcting Code*
MTBF	*Mean Time Between Failure*
MTTF	*Mean Time to Failure*
MTTR	*Mean Time to Repair*
PMI	*Project Management Institute*
POFOD	*Probability of Failure on Demand*
ROCOF	*Rate of Occurrence of Failures*
TMR	*Triple Modular Redundancy*

SUMÁRIO

1 INTRODUÇÃO

A cada vez mais as pessoas estão preocupadas em se certificarem em adquirir um produto de confiança, que possa lhe oferecer a segurança de estar sempre disponível para o uso quando precisarem. Dessa forma, não é diferente com os usuários de sistemas de software e clientes que desejam contratar uma empresa de software para construir um novo sistema. Isso se tem tornado uma preocupação comum, devido aos inúmeros tipos de falhas que um novo software implantado estar sujeito, como falhas em razão de entradas indevidas pelos usuários, perda de dados de um formulário de cadastro, incapacitação de finalizar uma determinada tarefa, etc. Sendo que, em virtude de certas falhas, às vezes é necessário colocar o sistema indisponível para fazer sua manutenção.

Os processos confiáveis de desenvolvimento de software foram criados para tomar todas as medidas e desenvolver um planejamento de todo o projeto de software, como forma de se desenvolver um sistema que seja confiável. Esse processo deve levar em consideração todos os requisitos de segurança e proteção necessária para o projeto que se quer desenvolver. Diante desses requisitos é feito todo um gerenciamento dos possíveis riscos que podem afetar a confiabilidade do sistema. São estudadas e analisadas formas de contenção e eliminação desses riscos, adotados técnicas e mecanismos de prevenção de defeitos e tolerância a falhas, como redundância e diversidade de componentes.

O desenvolvimento do projeto, ou melhor, a codificação do sistema vai consistir em colocar em prática todas as técnicas que foram incluídas no planejamento do sistema que estar sendo construído. É durante essa etapa que será analisado se os métodos escolhidos são realmente os mais adequados e se novos devem ser incluídos. As funcionalidades do sistema serão desenvolvidas e junto com elas, é aplicado o gerenciamento de riscos para que se garanta o perfeito funcionamento das funções do sistema.

Quando o sistema completo é terminado, ele deve ser submetido a testes de confiabilidade para que se possa medir o quanto confiável é o software desenvolvido. São utilizadas métricas, como forma de medir sua taxa de falhas, a probabilidade de falhas sob uma demanda, sua disponibilidade, entre outras. Desse modo, é estipulada a confiabilidade do sistema e analisado se ela é correspondente

com o que se esperava. Caso afirmativo, o sistema já pode ser implantado no ambiente de uso e pode-se dizer que ele alcançou a confiabilidade pretendida.

1.1 Motivação

Em virtude das constantes falhas que os sistemas de software estão sujeitos, é preciso adotar um processo confiável que garanta toda a confiabilidade que se deseja obter para um determinado sistema. Esse processo deve empregar mecanismos de tolerância à falha, dispositivos de autoverificação de erros e técnicas de prevenção de defeitos. É preciso que o processo de desenvolvimento garanta a confiança de que o sistema vai estar disponível para uso, de forma a fornecer todas as suas funcionalidades de forma integral e executadas de forma correta.

1.2 Objetivos

1.2.1 Objetivo Geral

O objetivo dessa monografia é propor um processo de desenvolvimento de software que seja confiável e garanta a confiabilidade necessária para um sistema. Esse processo deve conter um planejamento de gerenciamento de risco e utilizar meios de garantir o emprego de todos os requisitos de confiabilidade necessária ao sistema. O processo é baseado em processos de desenvolvimento existentes e mistura técnicas comuns a projetos de sistemas seguros e confiáveis.

1.2.2 Objetivos Específicos

a) Mostrar as propriedades da confiança necessária para se adquirir um sistema confiável.
b) Especificar os requisitos de confiança e proteção adotados nos processos confiáveis de software.
c) Mostrar como é feito o gerenciamento de riscos, passo a passo, desde o planejamento até o controle e monitoramento dos riscos identificados.
d) Exibir quais as principais métricas utilizadas nos testes de confiabilidade de software, e as etapas necessárias para efetuar esse teste.

e) Propor um processo confiável de desenvolvimento de software que utiliza como base os requisitos de confiabilidade.

1.3 Organização do trabalho

O trabalho encontra-se dividido em 5 capítulos. No capítulo 1 é feita a introdução de todo o texto, bem como seus objetivos e suas subdivisões.

No capítulo 2, são relatados todos os requisitos de confiança e proteção necessária aos sistemas de software confiáveis. São descritas as propriedades da confiança, as técnicas para tornar o sistema tolerante a defeitos, e a relação que a disponibilidade tem com a confiabilidade do software. E também, as métricas utilizadas para se medir a confiabilidade dos sistemas.

O capítulo 3 mostra as técnicas e estratégias empregadas para melhorar a confiabilidade do sistema. Como utilizar um processo confiável de software, adotar mecanismos de tolerância a falhas, e uma arquitetura que seja tolerante a defeitos. É relatado como é feita a gestão de risco, como medir o número de falhas em relação à demanda de serviços e como são feitos os testes que estipulam a confiabilidade do software.

O capítulo 4 é uma proposta de um processo confiável de software. Esse processo consiste nas fases de planejamento e modelagem, desenvolvimento, testes e implantação. É empregado o gerenciamento de risco durante cada fase e os requisitos de confiança relatados nos capítulos anteriores. A fase de testes utiliza medições de confiabilidade baseadas nas métricas existentes e segue um processo definido por uma série de passos.

O capítulo 5 apresenta as conclusões tiradas desse trabalho, fazendo uma retrospectiva geral. É avaliado se os objetivos foram alcançados e quais são os trabalhos futuros referentes ao tema abordado.

2 CONFIANÇA E PROTEÇÃO DE SISTEMAS DE SOFTWARE

Os sistemas de computador com o passar dos anos têm-se tornado cada vez mais complexos, cheios de funcionalidades e mais propícios a falhas. Para que os usuários tenham confiabilidade em um sistema, eles têm observado o quanto o sistema está disponível, e se o mesmo é capaz de cumprir todas as suas funcionalidades e da forma certa. Este aumento da complexidade nos sistemas de software tem requerido uma maior atenção no que diz respeito à proteção contra falhas.

Neste capítulo será abordado como identificar se um sistema é confiável, através das propriedades da confiança, que servem para verificar o quão fidedigno é um sistema, cumprindo com suas funcionalidades perante seus usuários. Também, é relatada a relação entre disponibilidade e confiabilidade, sendo que, quanto maior o tempo que um sistema fica disponível, maior é o seu nível de confiança. Ver quais os princípios fundamentais para construir sistemas confiáveis, através da prevenção, detecção e recuperação de falhas. Além disso, são mostrados quais os requisitos de sistema para se obter um sistema que seja confiável pelos usuários e protegido em relação a falhas.

Outro tema que é abordado nesse capítulo, é sobre os sistemas críticos de segurança, que deve sempre executar uma operação de forma segura, de forma a não causar nenhum dano ao seu usuário. Uma parte de suma importância, desse capítulo, é a proteção contra falhas, que traz consigo uma especificação dos requisitos dirigida a riscos, uma especificação de segurança para minimizar a ocorrência de falhas, e os requisitos de proteção contra falhas do sistema. Há também uma especificação de confiabilidade, relatando as métricas que determinam a confiança do sistema, e os requisitos funcionais e não funcionais de confiabilidade.

2.1 Sistemas confiáveis

O Software já se tornou algo comum na vida das pessoas. Ele estar presente nos mais diversos aparelhos e equipamentos utilizados no dia a dia, como eletrodomésticos, eletrônicos, celulares, etc. Devido essa presença constante se faz

necessária uma maior preocupação com as falhas que possam vir a prejudicar a eficiência de suas funcionalidades.

O sistema deve estar sempre disponível, e garantir o cumprimento de forma correta das funções que ele se compromete a fazer. Dessa forma, ele garantirá confiança aos seus usuários, que esperam que o sistema esteja sempre disponível e pronto para executar suas tarefas. Portanto, sistemas confiáveis de software são aqueles dos quais os usuários vão sempre, ou quase sempre, contar com sua disponibilidade e que possuem prevenção contra falhas de sistema, e mesmo que ocorram falhas, são capazes de detectá-las e se recuperarem das possíveis falhas.

2.1.1 Propriedades da confiança

Existem quatro principais propriedades da confiança, que vão expressar o quanto o sistema é confiável. A disponibilidade é uma dessas propriedades, que serve para determinar o funcionamento do sistema, se ele está executando e seus serviços são de utilidade para o usuário. Já a confiabilidade, vai definir o quanto o sistema é capaz de corresponder nos serviços prestados aos usuários, se o sistema executa com exatidão as tarefas requeridas pelos usuários (SOMMERVILLE, 2011).

Outra propriedade da confiança é a segurança, que vai analisar a possibilidade do software causar algum dano às pessoas que o utilizam, ou ao ambiente. A proteção vai garantir a integridade do sistema, de forma que os dados do sistema não sejam danificados por uma falha ou invasão (SOMMERVILLE, 2011). Sendo essas, as principais propriedades da confiança: disponibilidade, confiabilidade, segurança e proteção, observadas na figura 2.1.

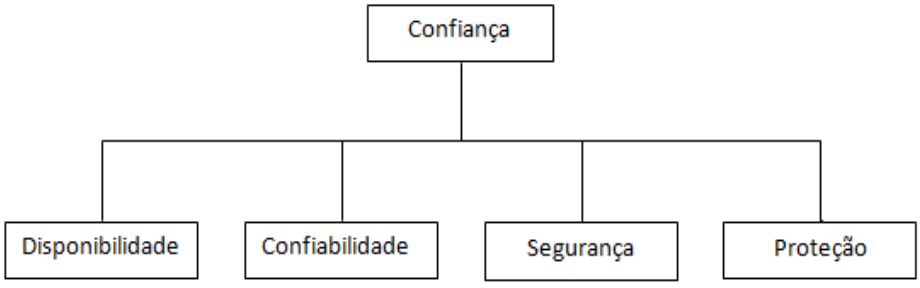

Figura 2.1: Principais propriedades da confiança

Fonte: SOMMERVILLE, 2011.

Além das propriedades citadas há também outras propriedades que servem para estimar o nível de confiança do sistema. Como a reparabilidade, que é a capacidade de reparar as falhas do sistema de forma rápida, sem que possa comprometer a utilidade do mesmo. Outra propriedade é a manutenibilidade, que é a possibilidade de se inserir novos requisitos ao sistema após ele já está em uso. O sistema também tem que ser tolerante a erros, de forma a detectar os erros de usuários e corrigi-los de forma a evitar falha de sistema (SOMMERVILLE, 2011).

Para desenvolver um sistema confiável, deve-se garantir que seja evitada a introdução de erros no sistema durante o seu desenvolvimento. Tem que ser adotados processos de verificação e validação, para descobrir erros. Além de se projetar mecanismos de proteção, e se atentar ao fato de se configurar e implantar o sistema de forma correta para o ambiente operacional (SOMMERVILLE, 2011).

2.1.2 Relação entre disponibilidade e confiabilidade

A disponibilidade e confiabilidade de um sistema estão profundamente relacionadas. A disponibilidade é a possibilidade de o sistema está pronto para uso e proporcionando serviços quando os usuários solicitarem. A confiabilidade de um sistema é a probabilidade de os serviços do sistema serem entregues, tal como definido na especificação do sistema (SOMMERVILLE, 2011).

Às vezes, a confiabilidade é mais importante que a disponibilidade para um determinado sistema, ou vice-versa. Se o usuário espera que o sistema esteja sempre executando e pronto para atender suas solicitações, então, a disponibilidade é um fator primordial, de forma que o sistema deve estar sempre disponível. Por outro lado, se a maior preocupação é com os danos causados pelas falhas de sistema, e se o software é capaz de se recuperar de eventuais falhas, então, a confiabilidade deve ser um ponto-chave para seus usuários.

De forma mais precisa, a confiabilidade é definida como a probabilidade de uma operação livre de falhas durante um tempo especificado, em determinado ambiente, para uma finalidade específica. A disponibilidade é a probabilidade de um sistema, em determinado momento, ser operacional e capaz de entregar os serviços solicitados (SOMMERVILLE, 2011).

A disponibilidade de um sistema é medida, além do número de falhas ocorridas, pelo tempo que o sistema leva para se recuperar dos defeitos que

causaram a falha (SOMMERVILLE, 2011). Supondo que um determinado sistema X leve pouco tempo para se recuperar de uma falha, então, é possível dizer que ele possui alta disponibilidade. No entanto, se esse sistema falha com determinada frequência, então, ele possui baixa confiabilidade. Sendo assim, é possível afirmar que os problemas de confiabilidade e disponibilidade do sistema são geralmente causados por falhas de sistema (SOMMERVILLE, 2011).

2.1.3 Tolerância a defeitos para construir sistemas confiáveis

A confiabilidade de um sistema tem relação direta com a possibilidade de ocorrência de um erro durante o seu período de uso. Por isso, sistemas confiáveis precisam incluir códigos redundantes para ajudá-los a se monitorar, a detectar estados errôneos e a se recuperar de defeitos antes que ocorram falhas (SOMMERVILLE, 2011). Dessa forma, quanto mais confiável é um software, a probabilidade de ocorrência de falhas se torna menor.

As falhas de sistemas são resultados de erros de especificação do próprio sistema ou de outros sistemas relacionados. Mas, a maioria das falhas acontece devido a erros do sistema, consequência dos defeitos contidos nele. Um defeito é definido como um desvio da especificação. O sistema está em estado de erro quando o processamento posterior a partir desse estado pode levar a um defeito. A falha vem ser a causa física ou *algorítmica* do erro (WEBER, 2003). A relação entre falha, erro e defeito é relatada com mais detalhes no modelo exposto de três universos conforme a figura 2.2.

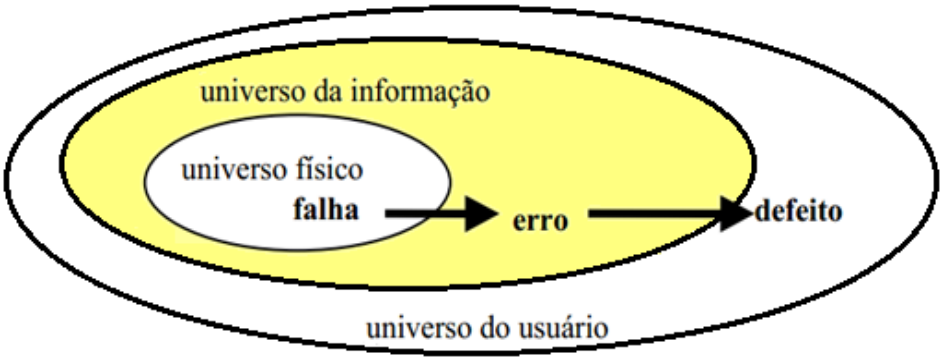

Figura 2.2: Modelo de três universos: falha, erro e defeito.
Fonte: Adaptado de WEBER, 2003.

Uma das abordagens para obter sistemas confiáveis é a prevenção de defeitos, que inclui entre outras coisas, evitar o uso de linguagens de programação tendente a erro, como uso de ponteiros, por exemplo. Devem-se usar técnicas de verificação e validação como auxílio para detecção e remoção de defeitos antes que o sistema seja colocado em uso. Adotar recursos de autoverificação em um sistema, além de usar módulos redundantes de sistemas, é uma forma de garantir tolerância a defeitos. A tolerância a defeitos vai certificar que os defeitos ocorridos no sistema não ocasionem em erros, ou que erros não se tornem em falhas do sistema (WEBER, 2003).

2.2 Requisitos de sistema para proteção e confiança

Nesse tópico serão abordados quais os requisitos de confiança e proteção utilizados para proteger o sistema contra falhas. Estes requisitos vão determinar a forma como o sistema se protege dos defeitos internos, estabilizando as falhas que possam vir a causar prejuízos ao sistema e ao ambiente.

2.2.1 Especificação de requisitos dirigida a riscos

Para descobrir quais os requisitos de confiança e proteção necessária a cada sistema, antes é preciso fazer uma análise dos riscos para o sistema e o ambiente. Essa análise deve levar em conta a possibilidade de eventos perigosos e os prejuízos causados na sua ocorrência. Nos sistemas críticos de segurança ou de proteção deve-se atentar a especificação desses requisitos, de forma a evitar acidentes, ou até mesmo expor vulnerabilidades do sistema (SOMMERVILLE, 2011).

O processo de especificação de requisitos dirigida a riscos abrange a identificação dos riscos enfrentados pelo sistema, fazer uma análise desses riscos para descobrir suas causas e procurar formas de redução dos riscos. A identificação dos riscos vai identificar os possíveis riscos para o sistema, que possam surgir a partir de interações do sistema com seu ambiente operacional. A análise e classificação de riscos irão considerar cada risco individualmente, de forma a eliminá-los quando não forem passíveis de ocorrerem. Uma decomposição dos riscos é feita para descobrir suas causas (erros de software, de hardware,

vulnerabilidades). Depois, a redução de riscos é aplicada para reduzir ou eliminar os riscos, aumentando a confiança do sistema (SOMMERVILLE, 2011). Os requisitos dirigidos a riscos estão especificados no esquema da figura 2.3.

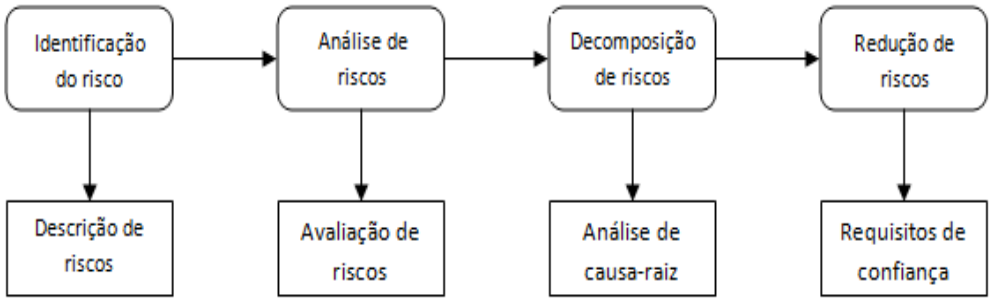

Figura 2.3: Especificação dirigida a riscos
Fonte: Adaptado de SOMMERVILLE, 2011.

2.2.2 Especificação de segurança para minimizar a ocorrência de falhas

Assim como a especificação de requisitos dirigida a risco está voltada para sistemas críticos de segurança, também está a especificação dos requisitos de segurança. Dessa forma, se faz necessário uma distinção entre perigo, que é algo que pode ou não resultar em morte ou ferimento de uma pessoa, e risco, que é a possibilidade do sistema entrar em um estado perigoso (SOMMERVILLE, 2011). Sendo assim, é possível dizer que a especificação de segurança se concentra nos perigos que possam surgir e nas suas causas.

A especificação de segurança segue basicamente os mesmos passos da especificação baseada em riscos, sendo organizada da seguinte forma: A identificação de riscos vai identificar os perigos que identificam os riscos que podem ameaçar o sistema; a análise de riscos vai fazer uma análise dos riscos para descobrir quais situações são mais perigosas ou mais prováveis; a decomposição de riscos, ou análise de riscos irá descobrir os eventos que podem ocasionar um perigo; e a redução de riscos, que conduzirá à identificação de requisitos de segurança, que podem garantir que perigos não surjam ou que não conduzam a um acidente ou, ainda, se ocorrer um acidente, que os danos causados sejam minimizados (SOMMERVILLE, 2011). A figura 2.4 mostra as três categorias de risco que podem ser usadas na avaliação de risco de sistemas de segurança crítica.

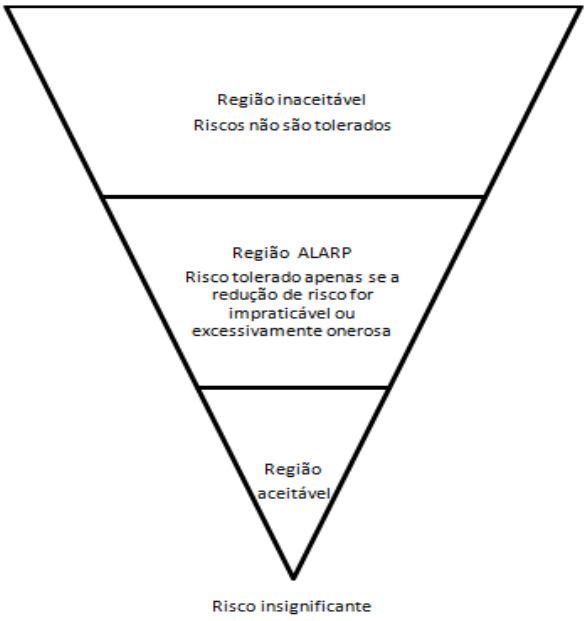

Figura 2.4: O triângulo de risco
Fonte: Adaptado de SOMMERVILLE, 2011.

2.2.3 Especificação de requisitos de proteção para sistemas

Os requisitos de proteção são parecidos com os requisitos de segurança. Esses requisitos vão definir os comportamentos inaceitáveis do sistema, em vez de definir a funcionalidade requerida (SOMMERVILLE, 2011). Eles se preocupam em verificar os problemas e tomar medidas para solucioná-los.

A especificação de requisitos dirigidos aos riscos pode ser utilizada para identificar os requisitos de proteção de sistema através de três estágios. Primeiro, é feita a análise preliminar de riscos, que consiste em escolher os melhores requisitos de proteção para o sistema. Depois, é feita a análise de riscos de ciclo de vida que tem como objetivo uma avaliação durante o desenvolvimento do sistema para analisar os riscos de acordo com as tecnologias adotadas. E por último, é feita uma análise de riscos operacionais que vai levar em conta a possibilidade de ataques de usuários ao sistema, através de suas vulnerabilidades (SOMMERVILLE, 2011).

2.3 Especificando confiabilidade

A confiabilidade do software vai depender além dos seus requisitos, dos requisitos do hardware e dos usuários desse sistema. É necessário pensar em

requisitos que possam suprir as falhas de software, ou que possam detectar e recuperar as falhas de hardware e erros do usuário. Assim, é preciso um acompanhamento da operação do sistema ao longo do tempo, para que se possa especificar o nível de confiabilidade necessário. Essa especificação pode ser feita com base na especificação dirigida a riscos, exibida anteriormente na figura 2.3.

A identificação de riscos vai identificar os tipos de falhas de sistema que podem levar a perda econômica (veja a lista de tipos possíveis de falha, mostrada na tabela 2.1). A análise de riscos serve para determinar os custos e as consequências resultantes dos diversos tipos de falhas de software, e selecionar as falhas com grandes consequências para analisar depois. Na fase de decomposição de riscos são examinadas as causas das falhas graves e possíveis de sistema, sendo que, talvez seja necessário voltar a essa fase durante o projeto e o desenvolvimento. Na etapa de redução de riscos é especificado o quanto o sistema é confiável para determinado tipo de falha, uma vez que, serão desenvolvidos os requisitos funcionais de confiabilidade (SOMMERVILLE, 2011).

Tabela 2.1: Tipos de falhas de sistema

Tipo de falha	**Descrição**
Perda de serviço	O sistema está indisponível e não pode oferecer seus serviços aos usuários. Pode ser dividida em perda de serviços críticos e perda de serviços não críticos, em que as consequências de uma falha em serviços não críticos são menores do que as de falha em serviço crítico.
Entrega incorreta de serviço	O sistema não oferece os serviços de forma correta. Novamente, pode ser especificada em termos de erros menores ou erros na entrega de serviços críticos e não críticos.
Corrupção de sistema/dados	A falha do sistema provoca danos ao próprio sistema ou a seus dados. Geralmente, acontece em conjunto com outros tipos de falhas, embora não necessariamente seja dessa forma.

Fonte: SOMMERVILLE, 2011.

2.3.1 Métricas para especificar confiabilidade

As formas de se medir a confiabilidade de um software pode ser tanto através da quantidade de serviços demandado como através da taxa de ocorrência de falhas. Além desses dois métodos, é possível medir o nível de confiabilidade de um sistema pela sua disponibilidade. A probabilidade de falha sob demanda, conhecida como POFOD[1], define a probabilidade de uma demanda por serviços de um sistema

[1] POFOD, do inglês *probability of failure on demand* é uma métrica para calcular a probabilidade de falha sob demanda.

ocasionar uma falha. A taxa de ocorrência de falhas, conhecida como ROCOF[2], determina o número de falhas de sistema que podem ser observadas por um período de tempo ou um número de execuções do sistema. A disponibilidade, conhecida como AVAIL[3], vai exprimir a capacidade de o sistema prestar seus serviços quando for solicitado (SOMMERVILLE, 2011).

A POFOD é mais utilizada em casos em que a falha sob demanda pode ocasionar numa grave falha de sistema. Já a ROCOF é mais usada em sistemas que trabalham com uma grande quantidade de operações, sendo que, as demandas devem ser feitas de forma regular e contínua. A disponibilidade é medida quando há uma perda de serviço do sistema. Essa medição é feita através do tempo para a reparação e recuperação da falha que ocasionou na indisponibilidade (SOMMERVILLE, 2011). A tabela 2.2 mostra os diferentes níveis de disponibilidade e seus significados.

Tabela 2.2: Especificação de disponibilidade

Disponibilidade	**Explicação**
0,9	O sistema está disponível 90% do tempo. Isso significa que, em um período de 24 horas (1.440 minutos), o sistema estará indisponível por 144 minutos.
0,99	Em um período de 24 horas, o sistema estará indisponível por 14,4 minutos.
0,999	O sistema estará indisponível por 84 segundos em um período de 24 horas.
0,9999	O sistema estará indisponível por 8,4 segundos em um período de 24 horas. Grosso modo, um minuto por semana.

Fonte: SOMMERVILLE, 2011.

2.3.2 Requisitos de confiabilidade

Os requisitos de confiabilidade se subdividem em requisitos não funcionais, e em requisitos funcionais. Os requisitos não funcionais vão determinar o número de falhas aceitáveis durante o uso normal do sistema, e são também conhecidos como requisitos de confiabilidade quantitativa. Os requisitos funcionais irão determinar as funções de sistema e de software que evitam, detectam ou toleram defeitos no software, evitando que esses defeitos gerem falhas de sistema (SOMMERVILLE, 2011). Ou seja, esses requisitos utilizam técnicas de verificação e recuperação de erro, além de mecanismos de tolerância a falhas.

[2] ROCOF, do inglês *rate of occurrence of failures* é uma métrica para calcular a taxa de ocorrência de falhas.
[3] AVAIL, do inglês *availability* é uma métrica que calcula a disponibilidade.

Os requisitos não funcionais de confiabilidade são especificações quantitativas de confiabilidade e disponibilidade calculadas a partir das métricas descritas no tópico anterior e são utilizadas principalmente em sistemas de segurança crítica (SOMMERVILLE, 2011). A vantagem de se utilizar essa especificação, é que a partir dela é possível demonstrar que existem diferentes tipos de falhas de sistema e que, às vezes, é caro atingir uma alta confiabilidade. Também, é possível identificar quando o sistema atinge o nível requerido de confiabilidade, para que se possam parar os testes de sistema. Além, de ser possível detectar se há ou não a real necessidade de um alto nível de confiabilidade.

A medida exata de confiabilidade que um sistema deve possuir vai depender entre outras coisas, do tipo de sistema e do orçamento a ser gasto na sua construção, uma vez que, altos níveis de confiabilidade requerem altos gastos. É necessário ter em mente prejuízos referentes a falhas de sistema para se determinar a confiabilidade do sistema (SOMMERVILLE, 2011). Um sistema indisponível por um determinado período de tempo pode significar além de prejuízo financeiro, a perda de clientes e comprometer a imagem da empresa. Entretanto, dependendo do tipo de sistema, é preciso avaliar se realmente é necessário que ele esteja disponível continuamente ou por períodos específicos do dia, adequando, dessa forma, sua disponibilidade (SOMMERVILLE, 2011).

Os requisitos funcionais de confiabilidade podem ser de três tipos diferentes: requisitos de verificação, requisitos de recuperação e requisitos de redundância (SOMMERVILLE, 2011). Os requisitos de verificação servem para garantir que entradas incorretas sejam detectadas antes de serem processadas pelo sistema. Os requisitos de recuperação têm com função recuperar o sistema das falhas, restabelecendo seus serviços após a falha. Já os requisitos de redundância vão garantir a redundância ao sistema, para que a falha de um só componente não prejudique o serviço de todo o sistema (SOMMERVILLE, 2011).

3 ENGENHARIA DE CONFIANÇA E ENGENHARIA DE PROTEÇÃO

Nesse capítulo, serão abordadas as técnicas e estratégias para se desenvolver sistemas confiáveis. Há uma descrição de como componentes redundantes e diversificados ajudam a aumentar a confiança de sistemas. Também, são relatadas as três abordagens que contribuem para o aumento da confiança tanto de sistemas críticos como não críticos, que são: a prevenção de defeitos, a detecção e correção de defeitos e a tolerância a defeitos. Será mostrado por que os processos confiáveis de software são de vital importância para se desenvolver software confiável. Entenderá como o uso de componentes e mecanismos redundantes na arquitetura utilizada, possibilita tolerância a defeitos.

Será relatado como é feito o gerenciamento dos sistemas para avaliação e prevenção de riscos, de forma a identificar os tipos de riscos que o sistema está propício, a classificação do risco e as técnicas utilizadas tanto para analisá-lo como para atenuar seus efeitos. No tópico seguinte, será visto como é feita a tolerância e recuperação de falhas de software, como o sistema identifica e se recupera de erros, os tipos de falhas de software possíveis de ocorrerem, as técnicas de tolerância a falhas, como alcançar dependabilidade, e quais são e qual a utilidade das medidas de confiabilidade. E por último, são relatados os testes que são feitos para se medir à confiabilidade do sistema.

3.1 Técnicas e estratégias para melhorar a confiança de sistemas

Os engenheiros de software adotam várias técnicas e estratégias para garantir que o sistema que eles produzem possua a maior confiabilidade possível. Eles adotam abordagens que possam prevenir a ocorrência de defeitos, e caso os defeitos ocorram o sistema possa detectá-los e corrigi-los, bem como também tolerar os defeitos impedindo a ocorrência de falhas. A redundância e diversidade garantem mais confiança ao sistema, de forma a garantir reposição em caso de falha e aumentando a chance de não repetição do mesmo erro. Já os processos confiáveis de software garantem que o sistema seja desenvolvido dentro dos padrões de confiabilidade. As arquiteturas tolerantes aos defeitos irão garantir que o controle seja assumido por mecanismos redundantes durante a ocorrência de alguma falha

de sistema. Desse modo, essas técnicas e estratégias são de grande importância para garantir que o software desenvolvido seja confiável.

3.1.1 Abordagens para aumentar a confiança dos sistemas

Existem diversas técnicas para se aumentar a confiança de sistemas, de forma a diminuir os riscos e os prejuízos causados por possíveis falhas. Sendo assim, essas técnicas englobam três abordagens que servem de complemento para o desenvolvimento de um software confiável, que são: prevenção de defeitos, detecção e correção de defeitos e tolerância a defeitos. A abordagem da prevenção de defeitos deve ser adotada logo na fase de projeto e permanecer durante o desenvolvimento do sistema, utilizando técnicas para prevenir erros tanto no projeto, como na programação do software, minimizando ao máximo o número de defeitos que possam surgir quando o sistema estiver em execução, e dessa forma, diminuindo as possibilidades de aparecimento de falhas.

A abordagem de detecção e correção de defeitos visa à utilização de processos de verificação e validação com o intuito de descobrir e extinguir os defeitos presentes no software, antes mesmo que ele seja colocado em operação. Essa abordagem deve ser aplicada com mais rigor, especialmente em sistemas críticos, que requerem uma extensa verificação e validação para que se possa descobrir o maior número possível de defeitos antes da implantação (SOMMERVILLE, 2011) e ter uma garantia de que o sistema é realmente confiável, para que seja colocado em operação. Já na tolerância a defeitos, o sistema deve estar pronto para detectar defeitos ou comportamentos inesperados durante sua execução, de forma, a evitar a ocorrência de falha. Isso inclui a utilização de processos de verificação interna enquanto o sistema está sendo executado. No entanto, quando é necessário alto nível de disponibilidade e confiabilidade deve-se utilizar uma arquitetura de sistema que seja tolerante a defeitos (SOMMERVILLE, 2011).

3.1.2 Redundância e diversidade

A redundância e a diversidade são requisitos essenciais para que se possa garantir confiabilidade em um software. A redundância é a garantia de reposição de

um componente caso ocorra alguma falha. A diversidade vai garantir que os componentes redundantes do sistema sejam de tipos diferentes, diminuindo a possibilidade de eles falharem pelo mesmo motivo.

Um exemplo típico de redundância utilizado tanto pelas empresas como pelas pessoas em geral, é o *backup* de seus dados, mantendo uma cópia de seus dados como segurança no caso da ocorrência de alguma falha, ou seja, nada mais são do que cópias redundantes dos dados que irão garantir a integridade desses dados em caso de falha. Em sistemas confiáveis de software podem ser usados componentes redundantes que executam a mesma função de outros componentes do sistema, que entrarão em ação na ocorrência de falha do componente principal. Caso esses componentes sejam de tipos diferentes, ou seja, diversificados, um defeito comum em componentes replicados não resultará em uma falha de sistema (SOMMERVILLE, 2011). Essa redundância também pode ser obtida através da utilização de códigos de verificação adicional que podem servir para a detecção de defeitos antes da ocorrência de falhas, e ainda, utilizar de mecanismos de recuperação para manter o sistema funcionando normalmente (SOMMERVILLE, 2011).

Utilizar diferentes sistemas operacionais em servidores redundantes é algo comum em sistemas de software confiáveis. Os servidores são designados de forma a substituir o servidor principal caso ocorra alguma falha. Os diferentes sistemas operacionais em servidores de diversos tipos vão garantir a diversidade, que servirá de proteção contra ataques que possam explorar as vulnerabilidades comuns a um determinado sistema ou tipo de servidor. Garantindo, dessa forma, diversidade e redundância, de forma a proporcionar similar funcionalidade de formas diferentes (SOMMERVILLE, 2011).

3.1.3 Processos confiáveis de software

Processos confiáveis de software são processos projetados para produzir softwares confiáveis (SOMMERVILLE, 2011). Em outras palavras, são processos especificados formalmente de forma a tomar todas as precauções para se desenvolver um software que passe toda confiabilidade que o seu usuário necessita. Um processo confiável consiste em uma série de aprovações que vão garantir a confiança do sistema, além, de ser todo documentado, cumprindo todas as

exigências necessárias para se desenvolver um sistema crítico. Desse modo, pode-se ter certeza, que ao adotar um processo desse tipo, o software produzido será menos propício a erros e, consequentemente, terá menos probabilidade de ocorrência de falhas. A tabela 3.1 mostra as principais características dos processos confiáveis de software.

Os processos confiáveis de software também utilizam redundância e diversidade para proporcionar confiabilidade, uma vez que, eles fazem uso de atividades diversificadas para atingir o mesmo objetivo, como por exemplo, as inspeções e testes, cuja finalidade é encontrar erros no programa. As atividades adotadas dependem do sistema a ser desenvolvido, sendo que, elas têm a finalidade de evitar a introdução de erros no sistema, detectarem os erros e removê-los mantendo as informações do processo atualizadas (SOMMERVILLE, 2011). Algumas dessas atividades consistem em revisar os requisitos para comprovar sua completude e consistência, gerenciar os requisitos de forma a manter sempre o controle de suas mudanças, criar um modelo matemático do software para analisar de forma mais detalhada os requisitos do sistema, fazer uma modelagem gráfica do sistema, inspecionar o projeto e o programa, analisar estaticamente o código-fonte, planejar e gerenciar testes (SOMMERVILLE, 2011).

Tabela 3.1: Características dos processos confiáveis

Característica de processo	Descrição
Documentável	O processo deve ter um modelo de processo definido que estabelece as atividades do processo e a documentação que deve ser produzida durante essas atividades.
Padronizado	Um amplo conjunto de padrões de desenvolvimento de software que abrange a produção e a documentação de software deve estar disponível.
Auditável	O processo deve ser compreensível para pessoas diferentes daquelas que participam do processo, as quais podem verificar se os padrões de processo estão sendo seguidos e fazer sugestões de melhorias ao processo.
Diverso	O processo deve incluir atividades de verificação e validação redundantes e diversas.
Robusto	O processo deve ser capaz de se recuperar de falhas de atividades individuais de processos.

Fonte: SOMMERVILLE, 2011.

3.1.4 Arquiteturas tolerantes a defeitos

Arquiteturas de sistemas tolerantes a defeitos são aquelas que possuem componentes e mecanismos redundantes que vão possibilitar que o controle do sistema seja permutado entre os componentes do mesmo (SOMMERVILLE, 2011). Entre os sistemas que utilizam esse tipo de arquitetura estão os sistemas críticos, como sistemas de comando e controle de aeronaves. Um exemplo típico de

arquitetura tolerante a defeito seria o uso de servidores replicados, que no caso da falha de um servidor na execução de uma funcionalidade outro servidor que possui a mesma tarefa assumiria o comando e o servidor que apresentou defeito seria descartado (SOMMERVILLE, 2011).

Apesar da abordagem de utilizar servidores replicados oferecer redundância, ela não dispõe a diversidade necessária para lidar com uma falha de software que afeta todas as versões de software ao mesmo tempo. Por isso, é preciso programar outros estilos diferentes de arquiteturas, como por exemplo, uma arquitetura de sistema de proteção. Esse tipo de arquitetura é adotado em sistemas de proteção que estão geralmente associados a um sistema de controle, como um sistema que controla os freios de um trem caso esse ultrapasse um sinal vermelho e não seja detectada sua desaceleração (SOMMERVILLE, 2011). Os sistemas de proteção monitoram o ambiente de forma independente e, se os sensores indicarem algum problema que o sistema controlado não seja capaz de lidar, então o sistema de proteção é ativado para parar o processo ou o equipamento (SOMMERVILLE, 2011).

Na figura 3.1 pode-se observar a relação entre o sistema de proteção (que faz o monitoramento do ambiente e dos equipamentos) e o sistema de controle, sendo que, quando os sensores do sistema de proteção detectam algum problema, ele emite comandos para os atuadores desligarem o sistema ou acionar outros mecanismos de proteção. A grande vantagem do uso desse tipo de arquitetura consiste no fato do software de sistema de proteção possuir uma função simples, em comparação ao sistema de controle, que é monitorar e assegurar que o sistema permaneça sempre em um estado seguro (SOMMERVILLE, 2011).

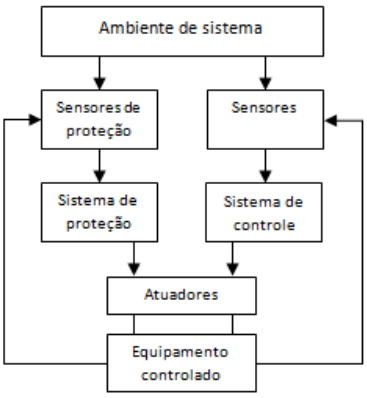

Figura 3.1: Arquitetura de sistema de proteção
Fonte: SOMMERVILLE, 2011.

Outro tipo de arquitetura tolerante a defeito é a arquitetura de automonitoramento, que consiste no próprio sistema monitorar seu funcionamento, e tomar as medidas necessárias para se recuperar das falhas detectadas (SOMMERVILLE, 2011). Esse monitoramento é feito por meio do cálculo de um valor de entrada em canais separados (o hardware e o software usados em cada canal, são diversificados), cujas saídas são comparadas por um comparador que identificará a ocorrência de falha caso os valores sejam diferentes. Quando a falha é identificada é gerada na linha de *status* de saída uma exceção de falha, e o controle é transferido para outro sistema. Observe esse esquema na figura 3.2.

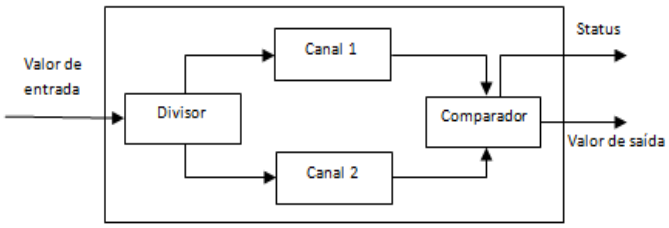

Figura 3.2: Arquitetura de automonitoramento
Fonte: SOMMERVILLE, 2011.

Sistemas TMR[4] são sistemas em que a unidade de hardware é replicada três vezes e suas saídas são comparadas por um seletor. Se duas ou mais entradas do comparador forem iguais, então esse valor é a saída, e caso uma das unidades falharem e produzir uma saída diferente das outras unidades, essa saída é ignorada. O gerente de defeitos pode tentar reparar a falha na unidade e, se não for possível, a unidade defeituosa é desativada, e o sistema continua operando somente com duas unidades (SOMMERVILLE, 2011). Veja o processo descrito, exposto em detalhes na figura 3.3.

[4] TMR, do inglês *triple modular redundancy*, que significa redundância modular tripla, é quando a unidade de hardware é replicada três vezes.

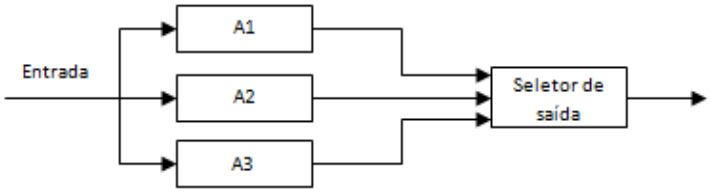

Figura 3.3: Redundância modular tripla
Fonte: SOMMERVILLE, 2011.

Uma abordagem que também pode ser adotada para softwares tolerantes a defeitos, e que é semelhante a que é utilizada em sistemas TMR, é a programação N-*version*, em que diversas versões do mesmo software executam em paralelo. Nela, o mesmo sistema é desenvolvido por várias equipes, cada equipe desenvolve uma versão, e cada versão é executada em um computador separado. Suas saídas são comparadas em um seletor de saída, e as saídas que forem inconsistentes ou não forem produzidas em tempo, são descartadas. Caso haja uma falha, pelo menos deverá haver três versões do sistema disponível para que duas possam ser consistentes com os resultados (SOMMERVILLE, 2011). Observe a ilustração na figura 3.4.

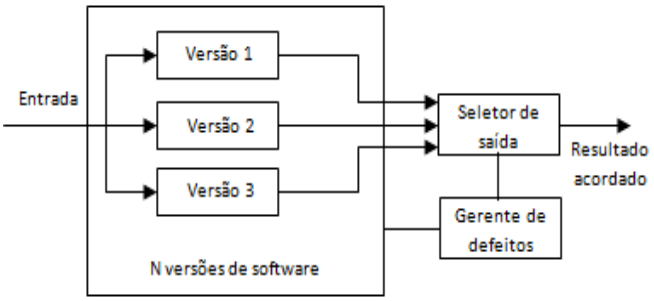

Figura 3.4 Programação N-version
Fonte: SOMMERVILLE, 2011.

3.2 Gerenciamento e avaliação de riscos de proteção

O gerenciamento e a avaliação de riscos são fatores fundamentais para a engenharia de proteção, sendo que, o gerenciamento deve ser debatido e decidido pela organização que deseja implantar o sistema, uma vez que, uma das suas preocupações é o balanço das perdas em relação aos custos de procedimentos de proteção. Cabe à gerência da organização decidir se é viável o custo da proteção ou

da exposição aos riscos por falta de procedimentos de proteção. Já a avaliação de riscos deve começar antes mesmo da decisão de adquirir o sistema e deve continuar durante todo o processo de desenvolvimento de sistema e depois que o sistema seja colocado em uso (SOMMERVILLE, 2011).

3.2.1 Gestão de risco

A gestão de risco compreende ações para identificar os riscos, descobrir suas fontes, a dimensão do risco, e adoção de formas de redução. É uma parte integrante do gerenciamento do projeto de software e contribui de forma direta para os objetivos da engenharia de sistemas. A identificação dos riscos consiste em ter uma dimensão dos riscos que podem vir a afetar o projeto do sistema. Ao identificar esses riscos é feito uma análise para descobrir as probabilidades de ocorrência e quais os danos que o risco proporcionaria ao projeto. Em seguida, devem-se buscar meios de prevenir o acontecimento desses riscos, ou até mesmo, minimizar seus danos caso seja impossível de evitá-los (SOMMERVILLE, 2011). Além disso, é preciso monitoramento constante para evitar que os mesmos riscos surjam novamente. Observe o processo adotado para a gestão de risco na figura 3.5.

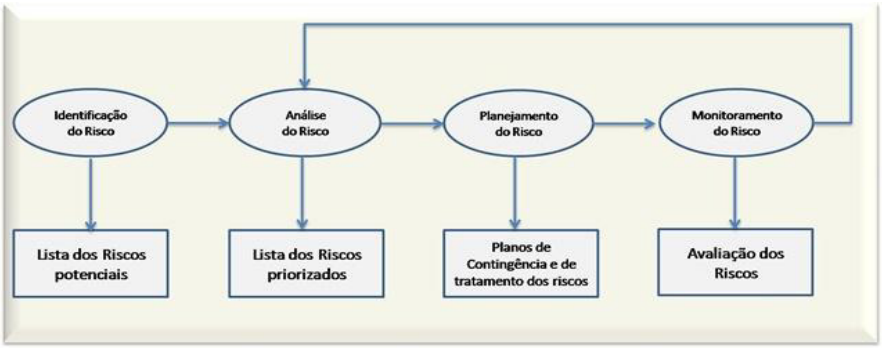

Figura 3.5: O processo de gerenciamento de riscos

Fonte: andrepitkowski.wordpress.com

3.2.2 Tipos de risco

Os tipos de risco estão divididos em categorias de riscos, e poderão ou não prejudicar o projeto do sistema de acordo com a forma que foi feita sua gestão de

risco. Segundo o PMI[5], as categorias de riscos são: riscos organizacionais, riscos de gerência de projeto, riscos técnicos, de qualidade ou de desempenho, e riscos externos (GOMEZ, 2010). Os riscos organizacionais estão ligados à gestão e política da empresa, como por exemplo, falta de verba, deixar de dar prioridade aos projetos, tempo e escopo inconsistente com a realidade da empresa. Já os riscos de gerência de projetos dizem respeito às falhas no gerenciamento do projeto, como ultrapassar fases do projeto na pressa de iniciar logo o desenvolvimento, atividades mal programadas, distribuição errada do tempo e dos recursos. Riscos técnicos, de qualidade ou de desempenho se referem à adoção de tecnologias da qual a equipe de desenvolvimento não domina, ou o uso de metas e performances muito complexas ou que não condizem com a realidade, e que pode prejudicar o desenvolvimento do projeto e a qualidade do produto final. E os riscos externos são aqueles que podem acontecer devido a fatores externos, como mudança das prioridades, catástrofes climáticas, guerras civis, etc. (GOMEZ, 2010).

3.2.3 Identificação do risco e técnicas de caracterização

A identificação do risco consiste em um levantamento de todos os riscos que possam vir a atrapalhar o andamento do projeto. É importante que toda a equipe participe dessa etapa, bem como, clientes e usuários finais do sistema, para que sejam determinadas todas as características dos possíveis riscos. Essa identificação pode ser feita através da revisão da documentação do projeto, por meio da documentação de projetos anteriores, e por meio de técnicas de coleta de informações, também conhecidas como técnicas de caracterização (ANDRADE, 2007).

As técnicas de caracterização são as seguintes: *check-list*[6], *brainstorming*[7], técnica *Delphi*[8], entrevistas, entre outras. A técnica de *check-list* consiste na elaboração de uma lista pelos *stakeholders*[9] com base na análise de problemas de projetos anteriores ou utilizando as habilidades dos gerentes e líderes de projetos, com o objetivo de marcar os riscos que podem comprometer o projeto em questão.

---

[5] PMI do inglês *Project Management Institute* (Instituto de Gerenciamento de Projeto) é uma organização profissional internacional sem fins lucrativos, que fornece métodos para gerenciamento de projetos.
[6] Check-list consiste de uma lista de riscos elaborada pelas pessoas envolvidas no projeto de software.
[7] Brainstorming é uma técnica adotada para listar os riscos do desenvolvimento do sistema.
[8] Delphi é uma técnica aplicada através de questionários para fazer o levantamento de riscos.
[9] Stakeholders, pessoas envolvidas direta e indiretamente no projeto de desenvolvimento de software.

Brainstorming, conhecida como tempestade de ideias, consiste em obter uma lista com diferentes ideias apresentadas em uma reunião entre todos os envolvidos no projeto (ANDRADE, 2007). Na técnica *Delphi* uma pessoa central, através de um questionário, solicita ideias sobre os riscos que podem afetar o projeto. Especialistas respondem o questionário anonimamente e devolvem para a pessoa central que resumirá as respostas e redistribuí-las aos especialistas, para que possam incluir novos comentários e novos riscos, até que um consenso entre eles seja alcançado. A entrevista consiste na elaboração de perguntas e a escolha dos entrevistados para que possam responder os questionários. É uma técnica simples e adotada por grande parte das empresas, devido necessitar de somente uma pessoa para elaborar as questões que pode ser o gerente de projetos ou uma pessoa especializada (ANDRADE, 2007).

3.2.4 Técnicas de análise de risco

As técnicas de análise de risco servem para definir as prioridades para os riscos que foram identificados, de forma a adotar certa cautela em relação aos riscos de alta prioridade. Essa análise é feita por meio de probabilidades dos riscos ocorrerem, o impacto que o risco provocaria no projeto, prazo e tolerância a risco, custo do projeto, cronograma, escopo e qualidade do projeto. A análise dos riscos é dividida em duas fases: análise qualitativa e análise quantitativa (ANDRADE, 2007).

A análise qualitativa tem como objetivo dar prioridade aos riscos, fazendo uma avaliação das probabilidades e impactos dos riscos, ou seja, é construída uma matriz de probabilidades. Essa análise é feita separadamente para todos os riscos identificados, onde é feita uma avaliação do risco por pessoas que já estão familiarizadas com esses riscos. Já a análise quantitativa define o efeito causado pelos riscos e atribui uma classificação numérica a eles, além de representá-los por meio de uma abordagem quantitativa para tomada de decisões (ANDRADE, 2007).

3.2.5 Técnicas de atenuação de risco

As técnicas de atenuação de risco servem para reduzir as ameaças aos objetivos do projeto (ANDRADE, 2007). Em outras palavras, é diminuir a probabilidade de ocorrência do risco ou seus impactos, produzindo uma situação

onde os riscos são resolvidos ou evitados. Para desenvolver essas atividades, podem-se utilizar técnicas de prototipação, simulação e desenvolvimento incremental.

Podem-se utilizar quatro ações sobre os riscos: evitá-los, contê-los, mitigá-los ou evadi-los (ANDRADE, 2007). Quando é decidido não dar procedimento ao projeto porque ele contém muitos riscos, isso nada mais é, do que uma forma de evitar os riscos. Para conter os riscos é preciso separar os recursos necessários para administrar os riscos caso eles ocorram. A mitigação é a atenuação do risco, minimizando seus impactos, ou até mesmo sua eliminação. Ao evadir o risco, os projetistas ignoram os efeitos que o risco pode trazer, não fazendo absolutamente nada para impedir sua ocorrência (ANDRADE, 2007).

3.2.6 Monitoramento e controle de risco

O monitoramento e controle de risco é um processo que consiste durante todo o ciclo de vida do projeto. Através dele, são encontrados novos riscos e observadas mudanças nos riscos já identificados durante a execução do projeto. Podem surgir novos riscos no decorrer do projeto, como também alguns podem diminuir ou aumentar seus impactos, além dos riscos já identificados após surgirem gerarem novos riscos. O processo de monitoramento e controle de riscos pode envolver a escolha de estratégias alternativas, execução de um plano de contingência ou alternativo, realização de ações corretivas e modificação no plano de gerenciamento de projeto (ANDRADE, 2007). O monitoramento dos riscos é fundamental para a gerência de risco funcionar de forma preventiva. Por outro lado, o controle dos riscos irá fazer uma avaliação da situação para identificar se há qualquer desvio no projeto de gerenciamento de risco daquilo que foi planejado (ANDRADE, 2007).

3.3 Tolerância e recuperação de falhas de software

É possível desenvolver softwares tolerantes a falhas que possam se recuperar dos possíveis erros de sistema, para isso, é preciso utilizar técnicas de tolerância e recuperação de falhas. Desse modo, será visto como a dependabilidade está estritamente ligada com a tolerância a falhas, qual sua importância e quais as técnicas que devem ser adotadas para se alcançar dependabilidade. Serão

mostradas as medidas usadas para se obter confiabilidade, e como mascarar falhas de software.

3.3.1 Dependabilidade

Em Engenharia de confiança e proteção de software, a palavra dependabilidade significa a qualidade do serviço fornecido por um determinado sistema e a confiança depositada no serviço fornecido (WEBER, 2003). Em outras palavras, expressa que a confiança do sistema está sempre disponível, executando suas funcionalidades de forma correta, sem a menor interferência de toda e qualquer tipo de falha. Por isso, alcançar dependabilidade é o objetivo principal da tolerância a falhas, para que se possa construir um sistema confiável.

A dependabilidade possui algumas medidas principais que servem como de atributos para alcançá-la, são elas: confiabilidade, disponibilidade, segurança de funcionamento, segurança, manutenibilidade, testabilidade e comprometimento do desempenho (WEBER, 2003). A confiabilidade como já foi dito anteriormente, é a garantia da prestação do serviço e que a inoperabilidade do sistema não comprometa a qualidade dos serviços prestados. Já a disponibilidade diz respeito à capacidade do sistema está operando e pronto para atender a requisição dos seus usuários de forma intermitentemente durante seu período de funcionamento.

O comprometimento do desempenho está relacionado com o desempenho do sistema, que no caso de falhas pode ser comprometido, sendo que, mesmo que o sistema continue funcionando, há uma perda de desempenho na prestação de seus serviços. A manutenibilidade é a facilidade de se realizar a manutenção do sistema de forma rápida e que o mesmo seja colocado em operação o mais rápido possível. A segurança de funcionamento garante que o sistema funcione de forma segura, ou que ele possa ser desligado quando colocar em risco outros sistemas ou pessoas que dependam dele (WEBER, 2003). A segurança protege o sistema contra falhas e ataques maliciosos de forma a manter a privacidade, autenticidade e integridade dos seus dados. E a testabilidade diz respeito à capacidade do sistema ser testável, realizar testes em seus atributos internos de forma a garantir sua manutenibilidade, e assim, reduzir a indisponibilidade do sistema devido a reparos (WEBER, 2003). Acompanhe os principais atributos de dependabilidade expostos na tabela 3.2.

Tabela 3.2: Atributos de dependabilidade

Atributo	Descrição
Confiabilidade *(reliability)*	Capacidade de atender a especificação, dentro de condições definidas, durante certo período de funcionamento e condicionado a estar operacional no início do período.
Disponibilidade *(availability)*	Probabilidade de o sistema estar operacional num instante de tempo determinado; alternância de períodos de funcionamento e reparo.
Segurança *(safety)*	Probabilidade do sistema ou estar operacional e executar sua função corretamente ou descontinuar suas funções de forma a não provocar danos a outros sistemas ou pessoas que dele dependam.
Segurança *(security)*	Proteção contra falhas maliciosas, visando privacidade, autenticidade, integridade e irrepudiabilidade dos dados.

Fonte: WEBER, 2003.

3.3.2 Medidas de confiabilidade

As medidas de confiabilidade são fornecidas especialmente para sistemas complexos, que requerem medidas estatísticas para definir sua taxa de defeitos, seu tempo médio de falha, o tempo médio de reparo e o tempo médio entre duas falhas (Veja a tabela 3.3). A taxa de defeitos é dada por falhas por unidade de tempo e varia com o tempo de vida de cada componente (WEBER, 2003). Na figura 3.6 pode ser observado que a taxa de defeitos é alta em softwares que ainda se encontram no início da fase de testes, e vai decrescendo até que o software seja colocado em operação. Durante sua vida útil, o software apresenta uma taxa de erro constante, até que, sua plataforma de hardware se torne obsoleta ou ele venha a precisar de alguma alteração, fazendo com que sua taxa de erros volte a crescer (WEBER, 2003).

Tabela 3.3: Medidas de confiabilidade

Medida	Descrição
Taxa de defeitos – failure rate, hazard function, hazard rate	Número esperado de defeitos em um dado período de tempo, assumido um valor constante durante o tempo de vida útil do componente.
MTTF – mean time to failure	Tempo esperado até a primeira ocorrência de defeito.
MTTR – mean time to repair	Tempo médio para reparo do sistema.
MTBF – mean time between failure	Tempo médio entre as falhas do sistema.

Fonte: WEBER, 2003.

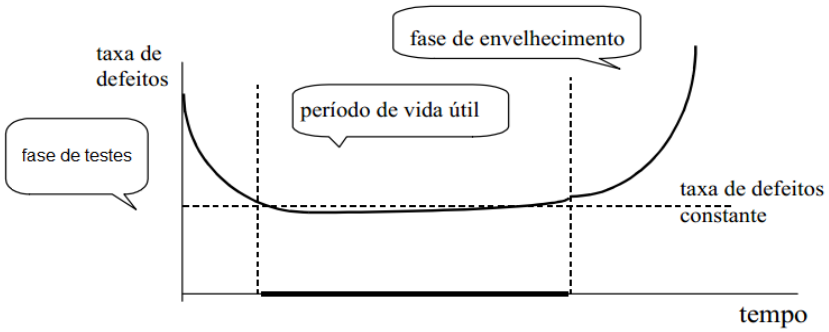

Figura 3.6: Curva da banheira
Fonte: Adaptado de WEBER, 2003.

3.3.3 Técnicas de tolerância a falhas

As técnicas de tolerância a falhas garantem o funcionamento do sistema mesmo que ocorra alguma falha. Por isso, elas utilizam componentes redundantes adicionais ou algoritmos especiais que irão preservar a disponibilidade e confiabilidade do sistema (WEBER, 2002). Essas técnicas estão divididas em duas categorias disjuntas, que pode ser o mascaramento, em que as falhas do sistema não irão se manifestar como erros, ou através da detecção, localização e reconfiguração, que é principalmente utilizada em sistemas de tempo real crítico (WEBER, 2002).

Alguns autores possuem sua própria classificação das técnicas de tolerância a falhas. Uma das técnicas mais conhecidas é a de quatro fases de aplicação (ANDERSON; LEE, 1981 apud WEBER, 2002, p. 17) descrita na tabela 3.4. A primeira fase é a detecção de erros que consiste em detectar as falhas latentes, que por ventura ainda não se manifestaram. Em seguida, é feito o confinamento e avaliação do erro de forma a limitar seus danos e evitar que ele se propague para outras partes do sistema (WEBER, 2002). A recuperação do erro consiste sair do estado incorreto para um estado livre de falhas e acontece utilizando técnicas de recuperação por retorno ou técnicas de recuperação por avanço (veja com mais detalhes na tabela 3.5). Na fase de tratamento de falhas, cada falha é analisada, individualmente, de forma a localizar a origem do erro, reparar a falha removendo os componentes danificados e recuperar o funcionamento de todo o sistema (WEBER, 2002).

Tabela 3.4: Quatro fases de Anderson e Lee

Fases	Mecanismos
Detecção de erros	Replicação Testes de limites de tempo Cão de guarda (*watchdog timers*) Testes reversos Codificação: paridade, códigos de detecção de erros, *Hamming*, ... Teste de razoabilidade, de limites e de compatibilidades Testes estruturais e de consistência Diagnóstico
Confinamento e avaliação	Ações atômicas Operações primitivas auto encapsuladas Isolamento de processos Regras do tipo tudo que não é permitido é proibido Hierarquia de processos Controle de recursos
Recuperação de erros	Técnicas de recuperação por retorno *(backward error recovery)* Técnicas de recuperação por avanço *(forward error recovery)*
Tratamento da falha	Diagnóstico Reparo

Fonte: ANDERSON; LEE, 1981 apud WEBER, 2002, p. 17.

As quatro fases de Anderson e Lee não englobam o mascaramento de falhas, que consiste de uma técnica à parte. Nessa técnica, a falha não se manifesta como erro, garantindo que o sistema exiba a resposta correta, mesmo na presença de falhas. Sendo assim, o sistema não entra em um estado errôneo, e dessa forma, os erros não precisam ser detectados, confinados e recuperados (WEBER, 2002). A tabela 3.6 mostra alguns dos mecanismos utilizados para mascaramento de falhas.

Tabela 3.5: Técnicas de recuperação

Técnica	Definição	Características	Implementação
Forward error recovery	Condução a novo estado ainda não ocorrido desde a última manifestação de erro	Eficiente, mas danos devem ser previstos acuradamente	Específica a cada sistema
Backward error recovery	Condução a estado anterior	Alto custo, mas de aplicação genérica	Pontos de verificação (*checkpoints*), pistas de auditoria (*audit trails*), ...

Fonte: WEBER, 2003.

Tabela 3.6: Mecanismos para mascarar falhas

Mascaramento de falhas	Redundância de hardware (replicação de componentes) Codificação: ECC (código de correção de erros) Diversidade, programação diversitária (n-versões) Blocos de recuperação

Fonte: WEBER, 2003.

3.4 Testes de estimação de confiabilidade

O objetivo dos testes de confiabilidade, como o nome mesmo já diz, é medir o nível de confiança no sistema. Para que possa ser especificada a confiabilidade de um software, pode-se utilizar as métricas já mostradas na seção 2.3.1 (POFOD, ROCOF E AVAIL), que utilizam métodos quantitativos para se medir o nível de confiança de um sistema (SOMMERVILLE, 2011). O processo de medição de confiabilidade de um sistema envolve quatro fases que são ilustradas na figura 3.7.

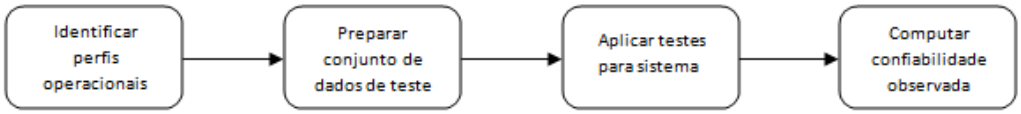

Figura 3.7: Medição de confiabilidade
Fonte: SOMMERVILLE, 2011.

O primeiro estágio a ser feito na medição de confiabilidade é identificar os perfis operacionais do sistema que se deseja testar. Essa identificação mostrará como o sistema será utilizado na prática, ou seja, são estabelecidas as classes de entradas de sistema e a probabilidade que essas entradas ocorram com uso normal (SOMMERVILLE, 2011). Para estabelecer o perfil operacional são estudados sistemas já existentes, do mesmo tipo do sistema que se estar desenvolvendo, para que, dessa forma, se possa compreender como esses softwares são utilizados na prática (SOMMERVILLE, 2011).

Após ser estabelecido o perfil operacional do sistema é preparado um conjunto de dados de testes de acordo com esse perfil identificado. Esses dados são criados utilizando a mesma distribuição de probabilidade que os dados de testes para os sistemas que foram estudados para se traçar o perfil operacional. O sistema é testado utilizando os dados de teste, de forma a contabilizar o total de falhas e os tipos de falhas que ocorreram, além, do tempo de cada falha. Diante do relatório completo de falhas, é possível, então, computar a confiabilidade de software e encontrar o valor apropriado de métrica de confiabilidade (SOMMERVILLE, 2011).

Esse processo é também conhecido como teste estatístico, e tem a finalidade de determinar a confiabilidade do sistema (SOMMERVILLE, 2011). No entanto,

possui algumas dificuldades para sua aplicação prática, como a dúvida se o perfil operacional gerado a partir de outros sistemas em uso, realmente reflete a forma como será usado o sistema que está sendo desenvolvido. Outra dificuldade é a alta elevação do orçamento devido à geração de um grande número de dados de teste. Além, de incertezas em relação à confiabilidade especificada estatisticamente ao sistema, e a identificação de ocorrência de falhas no sistema (SOMMERVILLE, 2011).

Algumas soluções são adotadas para que se possam contornar as dificuldades, como por exemplo, o uso de um gerador de dados de teste, de forma a automatizar a geração de dados de teste requeridos para medir a confiabilidade. Pode-se utilizar em conjunto com os testes estatísticos a injeção de defeitos, que vai inserir erros no programa que vão levar aos defeitos e falhas associadas (SOMMERVILLE, 2011). Dessa forma, é analisada se a causa da falha foi algum dos erros adicionados ao programa. Após, uma análise estatística da associação das falhas com os erros inseridos pode-se comprovar que os defeitos que surgem são compatíveis com os defeitos injetados.

Os testes estatísticos muitas vezes vão determinar que a confiabilidade do sistema estivesse muito abaixo daquilo que se espera e, no entanto, é preciso fazer reparos para melhorar essa situação. Depois de feitas as devidas modificações, são necessárias que se apliquem novamente os testes, e esse processo pode-se repetir por diversas vezes até que se atinja o nível de confiabilidade desejado (SOMMERVILLE, 2011). Dessa forma, a confiabilidade do sistema tende a melhorar ao longo do tempo e, quando não for possível alcançar a confiabilidade pretendida, devem-se ajustar os requisitos.

4 PROPOSTA PARA UM PROCESSO CONFIÁVEL DE SOFTWARE

Um dos desafios de se produzir um software confiável é a escolha de um processo que vai viabilizar a confiabilidade desejada. Esse processo deve garantir que todos os requisitos de confiança e proteção sejam seguidos à risca e colocados em prática. A especificação deve conter todos os requisitos necessários para garantir a confiabilidade do software, como por exemplo, descrever a medida de confiabilidade e disponibilidade exata que o sistema precisa. Devem ser adotados processos de verificação e validação como meio de acompanhar a aplicação dos requisitos de confiança e verificar se estão sendo empregados de forma correta.

Além de verificar a aplicação dos requisitos é preciso utilizar de técnicas que previne, detecta e corrige a ocorrência de defeitos. Elas são essenciais para evitar a ocorrência de falhas, sendo também, um dos mecanismos para evitar o aparecimento de falha. Os mecanismos de recuperação e tolerância a falhas envolvem, por exemplo, a utilização de redundância e diversidade, que na ocorrência de uma falha em um componente, o serviço não é comprometido por completo. Deve ser feito também, o gerenciamento de riscos, que envolve todo um planejamento, levantamento e identificação dos possíveis riscos, de forma a analisá-los para que se possa planejar como eles serão tratados pela equipe de desenvolvimento do software. Além, de fazer o monitoramento constante do sistema, de forma a controlar e tomar iniciativas para reduzir os riscos.

Durante todo o processo de desenvolvimento de software, como também, durante sua implantação, deve ser feito testes para medir sua confiabilidade. Antes da realização do teste é preciso fazer uma descrição do perfil operacional do sistema e levantar um conjunto de dados de teste que serão aplicados juntamente com as métricas de confiabilidade para se calcular a confiança do software. O resultado deve ser avaliado de acordo com o que foi definido na especificação de requisitos. Caso seja divergente, devem-se adotar medidas que melhorem a confiabilidade do sistema ou reaver os requisitos especificados.

Um processo confiável de produção de um software deve começar com uma boa especificação dos seus requisitos e adotar as melhores técnicas e mecanismos para implantar esses requisitos ao sistema. Além de aplicar os requisitos conforme foram especificados é preciso fazer o gerenciamento de risco, com o intuito de prever e prevenir danos que possam ser causados por prováveis riscos. Após o

acompanhamento e controle de riscos, então, é medida a confiabilidade do software, no intuito de descobrir se a confiabilidade atingida está de acordo com a que foi especificada para uso e aplicação do software. Se caso a confiança desejada não tenha sido alcançada, faz-se novamente a especificação de novos requisitos e todo o processo se repete. Quando o sistema atinge a confiança que dele se espera, então, o mesmo já está pronto para ser colocado em uso. Veja como esse processo ocorre através da figura 4.1. Como pode ser observado, todo o processo está dividido nas fases planejamento/modelagem, desenvolvimento, testes e implantação/manutenção do sistema.

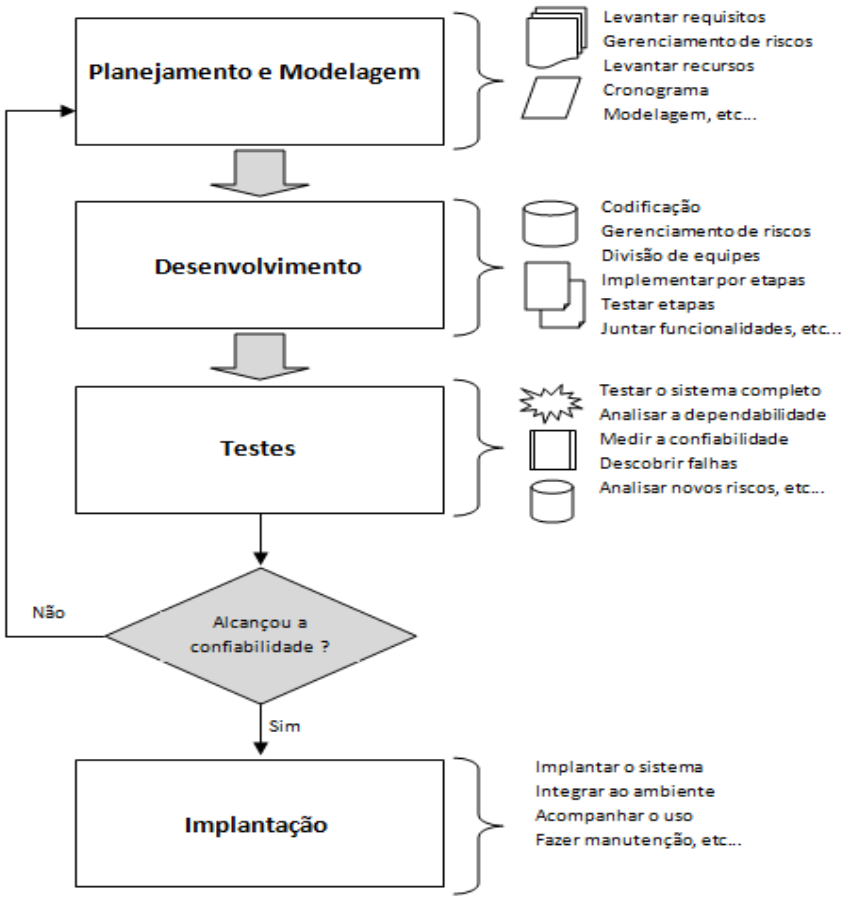

Figura 4.1: Visão geral do processo de software

4.1 Planejamento e modelagem

Essa fase consiste no planejamento de todo o processo de desenvolvimento do software, inclusive no levantamento dos seus requisitos. O planejamento vai descrever todas as tarefas que devem ser feitas pela equipe para incluir os

requisitos ao projeto. Deve ser feito o gerenciamento de riscos, para especificar os riscos inerentes ao projeto, como também a especificação orçamentária demandada durante seu desenvolvimento. São levantados todos os recursos a serem consumidos e as funcionalidades a ser produzidas, e, é feito o cronograma completo para a finalização do projeto. Modelos são criados para que se possa ter uma melhor dimensão de como os requisitos de software serão incorporados ao projeto. Dessa forma, pode-se ter uma representação da engenharia do software que será construído.

4.1.1 Especificação de requisitos

Os requisitos do sistema a ser desenvolvido devem ser especificados de forma clara e completa para que futuramente isso não possa refletir como futuros problemas no desenvolvimento do software. Esses requisitos vão incluir técnicas de verificação e recuperação de erro e mecanismos de tolerância a falhas, que farão parte dos requisitos funcionais do sistema. Como requisito de verificação a técnica proposta é verificar todas as entradas do sistema através da construção de padrões do uso do software. As entradas do sistema devem ser comparadas com esses padrões, de forma a verificar se as entradas não vão levar o sistema a um estado de erro, e só depois de se certificado é que a entrada do usuário é liberada para o sistema. Um esquema do uso de padrões de entradas é exposto na figura 4.2.

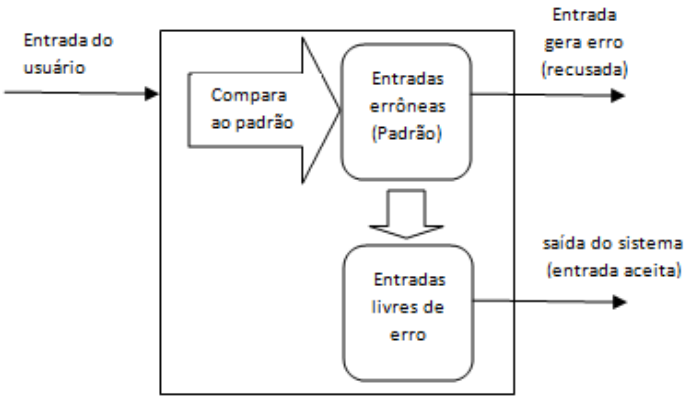

Figura 4.2: Verificando entradas com o uso de padrões

Os requisitos de recuperação vão envolver recuperação de erros e falhas que ocorrerem no sistema. A recuperação de erro pode ser feita através de pontos de

verificação que são salvos constantemente durante a utilização do sistema. Quando um erro ocorre o sistema recupera o estado anterior ao erro para que o usuário possa submeter novamente a operação que ocasionou o erro ou possa dar procedimento a outra operação diferente. Esse procedimento é descrito de forma mais detalhada na figura 4.3.

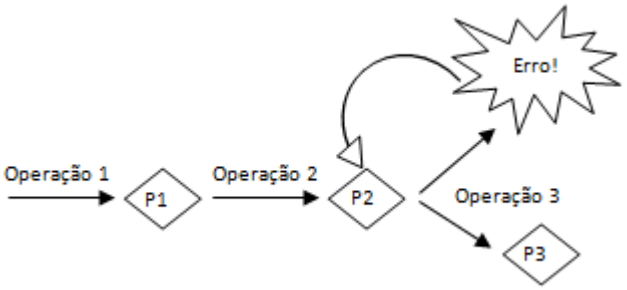

Figura 4.3: Recuperação de erro através de pontos de verificação

Os requisitos de redundância vão garantir através do uso de mecanismos de tolerância a falhas a operação do sistema mesmo na ocorrência de falha. Podem ser utilizados componentes e mecanismos redundantes, como servidores replicados, que são chaveados e outro servidor assume o controle em caso de falha. O sistema pode ser desenvolvido para mais de uma plataforma, de forma a garantir que falhas comuns em uma plataforma não comprometam seu uso em uma plataforma diferente. Como por exemplo, um sistema multiplataforma que contém servidores replicados e diferenciados.

Um dos mecanismos adotados para tolerância e recuperação de falha é a redundância de alguns componentes, em que os componentes são replicados e substituídos quando o componente principal falhar. Os erros devem ser identificados através de códigos de detecção de erros antes que eles ocasionem uma falha. Deve ser feito uma avaliação dos erros encontrados de forma a isolar os processos que os originaram para que o erro não se espalhe. Em seguida, o sistema é direcionado para o estado anterior ao erro, pois devem ser adicionados pontos de verificação no sistema, que serão salvos como pontos de retorno, no caso de falha. E por último, as falhas encontradas devem ser tratadas e diagnosticadas para que se encontre a origem do erro e se possam fazer os reparos diretamente no código-fonte, para que o sistema não volte a falhar. Veja o esquema de detecção, avaliação, recuperação e tratamento de falha na figura 4.4.

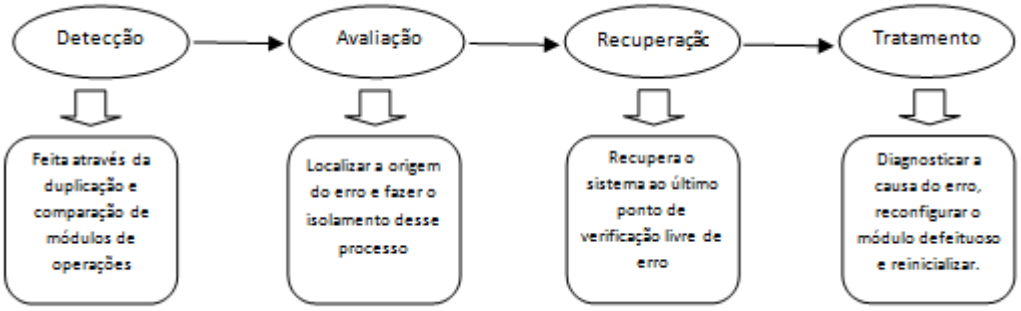

Figura 4.4: Tolerância e recuperação de falhas

4.1.2 Especificação dos requisitos de disponibilidade e confiabilidade

Além dos requisitos funcionais, é necessário especificar os requisitos não funcionais do sistema, como a quantidade de disponibilidade e confiabilidade que deve ser demandada ao software. A confiabilidade do sistema pode ser especificada de acordo com a demanda de serviços que o sistema terá quando colocado em uso. Deve-se observar se o fluxo de operações do sistema será muito grande, de forma a aumentar a confiabilidade de sistema. É calculada a probabilidade de o sistema falhar durante uma determinada demanda de serviços. Essa probabilidade é obtida através da quantidade de falhas observadas em relação a um número de serviços. A razão entre esses números mostra quantas falhas em média podem ocorrer para a demanda analisada. Veja um exemplo de cálculo da probabilidade de falha e sua taxa de ocorrência em relação à demanda na tabela 4.1.

Tabela 4.1: Probabilidade de falha e taxa de ocorrência

Demanda de serviços (*)	Quantidade de falhas	Probabilidade	Taxa de ocorrência
100	2	0,02	2/100
500	5	0,01	5/500
1000	10	0,01	10/1000

Para especificar a disponibilidade é necessário analisar o tempo que o sistema precisa estar disponível para os usuários. Dependendo do tipo de sistema que se estar desenvolvendo, não há a necessidade do mesmo ficar disponível 24 horas por dia e durante 7 dias da semana. Dessa forma, é preciso tomar cuidado

* Valores de demanda de serviços do sistema estipulados

para não demandar uma disponibilidade além da que o sistema realmente necessita. Por isso, deve ser feito um estudo para se avaliar a medida de disponibilidade que deve ser demandada ao sistema.

A disponibilidade do sistema pode ser medida através da razão entre o tempo que o sistema está disponível em relação ao tempo que ele deveria está em uso durante o decorrer do dia. É importante esse cálculo para saber se a porcentagem de tempo que o sistema fica disponível está de acordo com suas necessidades de uso. Na tabela 4.2 é demonstrado como fazer a medição da disponibilidade.

Tabela 4.2: Medindo disponibilidade

Tempo em uso (por dia)	Tempo disponível	Disponibilidade
24 horas	22 horas	0,9166
12 horas	11 horas e 30 minutos	0,9583
10 horas	8 horas e 30 minutos	0,85
8 horas	7 horas	0,875

4.1.3 Gerenciamento de riscos

O gerenciamento de riscos é importante para prever todos os riscos que o projeto de software pode estar passível. Por isso, deve ser feito um planejamento de gerenciamento de riscos para acompanhar o desenvolvimento do software e saber quais medidas tomar caso eles ocorram. O planejamento de gerenciamento de riscos deve ser feito antes do início do desenvolvimento do projeto e levar em consideração suas metas e limitações. O planejamento é feito com base em reuniões entre os engenheiros e desenvolvedores de software, levando em conta os requisitos do projeto de software e como o projeto está sendo gerenciado. Durante o planejamento, todos os possíveis riscos devem ser identificados e traçados planos de respostas para lidá-los, reduzindo ou até mesmo extinguindo seus impactos. Durante a elaboração do planejamento uma pequena equipe deve ficar responsável pelo monitoramento e controle de um determinado risco. Para que o planejamento funcione como esperado devem ser feitas revisões na sua especificação durante a fase de desenvolvimento. Com a inclusão de novos riscos que surgirem ou um novo plano de resposta mais eficiente para um risco que já foi especificado.

A identificação dos riscos será feita com base no estudo dos requisitos que foram levantados para o desenvolvimento do software, além da experiência de

projetos anteriores, para elucidar os possíveis riscos do projeto atual. Porém, a principal forma de levantamento de riscos deve englobar tanto a análise dos problemas ocorridos em projetos anteriores, como também, na experiência da equipe de projeto e desenvolvimento, que juntos devem se reunir e formar uma lista dos riscos possíveis. Também, como auxílio pode-se utilizar de entrevistas, já que se constitui numa técnica simples e altamente utilizada e acaba surtindo um bom efeito.

Os riscos levantados pela equipe devem ser revisados e analisados para checar se realmente estão de acordo com o projeto especificado. Pode-se fazer a elaboração de questionários que devem ser respondidos tanto pela equipe responsável pela elaboração do software como pelos clientes e usuários, para que se possa ter certeza da necessidade de sua inclusão no planejamento de riscos. Após a elaboração completa da lista de riscos é feita o cálculo da probabilidade do risco acontecer e então, uma prioridade é atribuída a ele, com base na avaliação de seus impactos.

A probabilidade do risco é calculada com base nos questionários aplicados, nas entrevistas, e nos dados levantados durante as reuniões. Depois de calculada a probabilidade de o risco acontecer e os impactos decorridos, ele é classificado numa escala de prioridades, que varia entre as categorias baixa, média e alta. O impacto que o risco produz vai depender dos objetivos do projeto que ele afeta, como custo, tempo de conclusão do projeto, diminuição do escopo ou qualidade, entre outros. Cada equipe de desenvolvimento pode estipular sua melhor forma de classificação de impactos dentro das categorias: baixo, moderado e alto. Uma classificação completa de riscos pode ser vista na tabela 4.3.

Tabela 4.3: Prioridade dos riscos relacionada as suas probabilidades e impactos

		Impacto do risco		
		Baixo	**Moderado**	**Alto**
Probabilidade	**0 a ≤ 10%**			
	10% a 40%			
	Acima de 40%			

Legenda: **Baixa Prioridade** **Média Prioridade** **Alta Prioridade**

Diante da lista de prioridade de risco, faz-se o planejamento das ações que devem ser tomadas para evitar sua ocorrência ou reduzir seus impactos. O planejamento de respostas aos riscos consiste na elaboração de metas que devem ser cumpridas com o objetivo de conter os danos oriundos dos riscos levantados. Para cada risco é considerado os impactos de maior proporção e levantadas estratégias que possam reduzi-los dentro do orçamento previsto no projeto. Dessa forma, para cada risco é desenvolvida uma estratégia que possa combater cada impacto por ele causado. Quando tiver elaborado uma ação de contingência para cada impacto do risco em questão, passa-se para o próximo risco da lista de prioridades, e assim por diante, até que para todos os riscos levantados seja elaborado um plano que possa conter suas ações.

A primeira estratégia adotada para lidar com os riscos levantados é a eliminação da sua possibilidade de surgimento e, se não for possível, planejar formas de reduzir seus impactos no projeto. Por isso, o plano de respostas aos riscos deve conter a especificação de cada risco, qual a melhor ação adotada pela equipe de software para combater seus impactos, o orçamento previsto para cada estratégia escolhida, as responsabilidades de cada atividade de combate, os riscos residuais que podem surgir após as ações de mitigação do risco, o tempo demandado por cada ação tomada.

A fase de monitoramento e controle de riscos vai servir, além de observar se as ações tomadas para lidar com eles estão de acordo com o que foi planejado no planejamento de respostas, se houve o surgimento de novos riscos ao projeto. Deve ser analisado se a resposta ao risco está surtindo efeito, ou é necessário se planejar novas ações. O controle de risco vai consistir em tomar ações corretivas para que o plano de resposta seja executado conforme foi planejado.

O monitoramento será feito através de um relatório constante das ocorrências surgidas durante a aplicação do plano de respostas aos riscos. Caso haja mudanças de escopo no projeto de desenvolvimento de software, um novo plano de resposta deve ser elaborado, com base no surgimento de novos riscos. Outra técnica que deve ser adotada durante o monitoramento é a auditoria dos processos realizados durante o controle de riscos. Pode ser feito um planejamento adicional para lidar com o risco, caso o planejamento proposto não for suficiente para combater os impactos que podem estar acima do que se esperava.

O monitoramento e controle de riscos vão conter planos de contorno para lidar com os riscos emergentes, ações corretivas para mudar a estratégia de conter o risco ou incluir novas estratégias, documentação e avaliação dos riscos ocorridos, eliminação do plano de risco dos riscos que não ocorreram, a criação de um banco de dados do risco. Desse modo, a preocupação com os riscos acaba se tornando um ciclo durante todo o decorrer do projeto, conforme é observado na figura 4.5. Assim, toda a fase de projeto, desenvolvimento e implantação do software será monitorada, no intuito de atingir a confiabilidade desejada ao sistema com a redução ao máximo dos riscos provenientes.

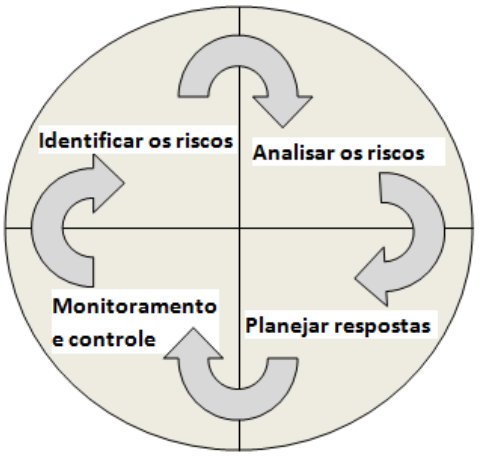

Figura 4.5: Ciclo do gerenciamento de riscos

4.2 Desenvolvimento do sistema

Após a conclusão da fase de planejamento e modelagem do sistema já pode ser iniciado seu desenvolvimento. Já foi feito todo um plano de como será o desenvolvimento, quais serão os mecanismos adotados para garantir a confiabilidade, juntamente com um estudo dos provenientes riscos e ações que devem ser tomadas. O sistema será desenvolvido com base nesse planejamento e nos modelos criados para garantir a confiabilidade que dele se espera.

Durante o desenvolvimento do sistema haverá o acompanhamento dos riscos e novos riscos podem ser levantados. Por isso, o gerenciamento de riscos estará presente também nessa fase do projeto. Os requisitos do sistema serão implantados de acordo com o que foi especificado no projeto, assim como seus requisitos de confiabilidade e disponibilidade.

A codificação será feita por etapas, sendo que, cada função poderá ser codificada separada, testada, e depois, implantada ao sistema. A equipe de engenharia e desenvolvimento de software será dividida em grupos, para que cada grupo fique responsável por desenvolver determinadas funcionalidades do sistema. Uma equipe ficará encarregada de fazer a integração das funções do sistema e fazer os testes de como elas funcionam em conjunto. Após a conclusão de todo o sistema, ele é testado por uma equipe de testes, que fará suas medições de confiabilidade.

4.3 Testes de Software

Os testes de software vão servir para que seja verificado o perfeito funcionamento das funcionalidades do sistema desenvolvido. Dessa forma, pode-se testar e medir o nível de confiabilidade atingido pelo novo software. A dependabilidade do sistema pode, também, ser avaliada com base na análise dos requisitos implantados. Assim, para avaliar a confiança do sistema, é preciso levar em consideração sua disponibilidade e a quantidade de falhas que pode ocorrer durante um determinado número de execuções.

4.3.1 Dependabilidade do sistema

Para que o sistema possua a dependabilidade que dele se espera, é preciso utilizar meios para prevenir falhas, aplicar técnicas de tolerância a falhas, fazer uma validação para verificar a presença de defeitos e removê-los, buscar estimativas sobre o surgimento de possíveis falhas e seus efeitos. Para fazer a verificação de defeitos pode-se utilizar tanto uma verificação estática quanto dinâmica. Na primeira, é feita uma análise do sistema através de ferramentas de análise, como por exemplo, análise do fluxo de dados, análise do comportamento do sistema, etc. Já a verificação dinâmica consiste na execução de testes de verificação, como verificação das possíveis entradas do sistema, por exemplo. A remoção dos defeitos encontrados pode ocorrer através de manutenção corretiva, para remover os defeitos que provocam erros no sistema, ou através da manutenção preventiva, que

vai eliminar os defeitos que ainda não são aparentes durante a execução do software.

Um levantamento do comportamento do sistema durante sua execução é feito como forma de prever o surgimento de possíveis falhas. Esse relatório servirá para identificar e classificar falhas e separá-las em categorias para que se possa analisar a forma como elas acontecem e quais os efeitos provocados. Essa análise será útil quando se forem calcular as medidas de confiabilidade do sistema, uma vez que, a previsão de falhas enfoca principalmente a modelagem e testes de avaliação que vão expressar as medidas de dependabilidade do sistema.

4.3.2 Medindo a confiabilidade do sistema

Antes que possa ser feito as medições de confiabilidade é necessário levantar o perfil operacional do sistema para que se conheça como o software será utilizado pelos usuários. De antemão do perfil operacional é preciso criar um conjunto de dados de teste para que os testes possam ser aplicados. Esses testes vão incluir as medidas de confiabilidade e disponibilidade, para que se possam ter medidas estatísticas para fazer a análise e estimação da confiança do software.

O perfil operacional do sistema vai ser levantado com base em softwares semelhantes produzidos anteriormente e que obtiveram sucesso na sua especificação de confiabilidade. Também, podem ser desenvolvidos protótipos que devem ser utilizados pelos futuros usuários do software em desenvolvimento, para que se possa analisar o perfil de uso desse sistema na prática. Pode-se fazer uma pesquisa com softwares que estão em uso e são semelhantes ao software produzido, para verificar o perfil de uso de cada funcionalidade. É feito um levantamento como os usuários utilizam o software para cada função levada em consideração, e a partir de várias amostras, se constrói um perfil de uso para cada atividade. Sendo assim, o perfil de todas as operações do sistema vai ser juntado para compor seu perfil operacional.

O perfil operacional vai dividir as possíveis entradas dos usuários em classes de entradas do sistema. Ao descobrir as classes de entradas é preciso determinar a probabilidade de ocorrência dessas entradas durante o uso normal do software. Diante das classes de entradas e suas probabilidades de ocorrerem já é possível fazer o levantamento dos dados que serão utilizados no teste.

Para construir o conjunto de dados de teste deve ser utilizado um gerador de dados de teste. Ele é ajustado para gerar dados que correspondam ao perfil operacional identificado. Os dados que se pretende testar devem refletir falhas comuns que comprometem o desempenho do sistema. São coletados dados que verificam a forma como os recursos do software serão utilizados e se o modo de utilização dessas funcionalidades irão gerar falhas no sistema. São levantados dados quanto à demanda de serviços do sistema, de forma a verificar seu desempenho em altas proporções de uso. Os testes devem incluir dados que verifique o comportamento do software em relação a falhas, se ele vai executar seus mecanismos de recuperação e tolerância a falhas conforme foi planejado. Também, devem ser levantados dados em relação à disponibilidade, para que se possa testar se sempre que os usuários requisitarem seus serviços, o sistema estará operando e pronto para atender suas requisições. Como alternativa na geração de dados de teste pode-se criar os comandos de geração de dados manualmente. Dessa forma, se teria um controle mais rígido dos dados gerados e mais condizentes com a realidade que se pretende testar.

Os testes de confiabilidade de software consistem na aplicação dos dados levantados a partir do perfil operacional do sistema para verificar suas respectivas saídas e a forma que suas operações foram executadas, de forma a comprovar a exatidão dos seus resultados com os requisitos de confiabilidade anteriormente definidos para o sistema. As falhas ocorridas durante os testes devem ser contabilizadas, assim como, devem ser registrados os tipos de falhas que ocorreram e o tempo dessas falhas. Os testes são realizados a partir de amostras estatísticas calculadas em cima de medições de confiabilidade.

Uma das primeiras medições que deve ser feita é calcular o tempo médio de ocorrência de uma falha e o tempo médio para reparar essa falha. Essas medidas são obtidas através de certo número de observações de casos de teste, e então, é estipulado um valor. Esses valores vão servir de parâmetro para se calcular o intervalo de tempo entre duas falhas consecutivas. Observe as fórmulas de aplicação dessas medidas e como calcular a disponibilidade do sistema a partir das medidas citadas. Essas medidas vão mostrar a quantidade de falhas do sistema. A figura 4.6 mostra como estimar o tempo médio entre falhas.

Tempo Médio entre Falhas = Tempo Médio de Falha + Tempo Médio para Reparar a Falha

Disponibilidade = Tempo Médio de Falha / (Tempo Médio de Falha + Tempo Médio de Reparo) x 100%

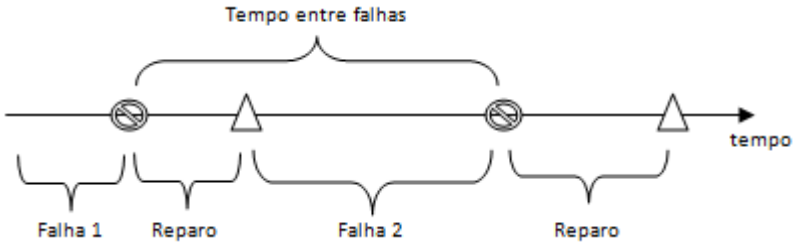

Figura 4.6: Tempo médio entre falhas

A confiabilidade do sistema em relação ao tempo ou quantidade de transações que a aplicação pode suportar sem ocasionar falha vai ser medida pela sua probabilidade de falhas em relação à demanda de serviços e pela sua taxa de ocorrência de falhas, conforme foi especificado nos requisitos de confiabilidade. A disponibilidade testará o tempo que é gasto para fazer reparos no sistema e colocá-lo em operação, com a restauração de todas as suas funcionalidades.

A medida de confiabilidade que será atribuída ao software vai depender da análise de todas as medições feitas nos testes. Por isso, essas métricas devem ser aplicadas repetidas vezes até que obtenha uma medida estatística para se especificar o quão confiável é o sistema. Os valores obtidos através das medidas devem ser passados para um programa específico que vai calcular a confiabilidade do sistema com base nesses dados. Algumas características do software podem ser observadas e analisadas para servir de critério como estimação de confiabilidade. É observado se o software é resistente a interrupções internas ou externas, preservando seus serviços aos usuários. O sistema deve se recuperar de um erro e restaurar o estado anterior, sem que haja perda de dados. A aplicação deve fornecer seus serviços de forma precisa mantendo um controle do fluxo de suas operações. É verificado se há interrupções nos serviços fornecidos pelo software devido a atualizações necessárias. É analisado se o sistema possui o mínimo de erros, prevendo somente pequenas atualizações futuras, estando pronto para ser colocado em uso. Com base na análise desses requisitos do sistema, ele pode ser tido como confiável, se houver uma previsão do seu funcionamento, conforme foi planejado.

4.4 Implantação do sistema

A implantação do sistema nada mais é do que sua disponibilização para uso. Depois de concluído seu desenvolvimento, o software é testado para garantir que todos os seus requisitos foram implementados da forma que foram especificados. O teste de confiabilidade do sistema é feito para saber se foi atingida a confiança que se esperava, então, o sistema é implantado no ambiente de uso.

No entanto, a utilização do sistema necessita de acompanhamento constante para se certificar que algumas falhas não passaram despercebidas e vão ocasionar erros futuramente. Por isso, a equipe continua prestando suporte técnico para que as devidas alterações no funcionamento do sistema sejam notadas assim que ocorrerem e que possam ser tomadas as devidas medidas de correção. Essas ações podem consistir tanto em pequenos reparos, como até a manutenção completa do sistema, dependendo da análise da falha.

O gerenciamento de risco precisa ser posto em prática mesmo na fase de implantação do sistema. Pois, como o objetivo de todo o processo é obter um software que possa ser confiável, é preciso tomar todo o cuidado para que nenhum risco de falha passe despercebido pela equipe responsável pelo desenvolvimento do projeto. Desse modo, mesmo durante a implantação do sistema, novos riscos podem surgir e com eles a necessidade de futuras manutenções no sistema.

A manutenção do sistema faz parte da sua fase de implantação. Uma vez que, o software precisará de acompanhamento e possivelmente modificações no seu escopo ou nos seus requisitos. Pode haver a necessidade de inclusão de novas funcionalidades, e com isso, é feito novamente o levantamento de requisitos e todo o processo pode ser iniciado novamente, como forma de garantir que a inclusão de uma nova função não comprometa o restante do sistema que já se encontra em funcionamento. Dessa forma, deve ser feito todo um acompanhamento, com a criação de um novo projeto específico para a função a ser desenvolvida, tendo como base o projeto completo do sistema. Os riscos têm que serem analisados e novos testes serão feitos, até que se tenha a certeza que a confiabilidade do software será mantida e possam-se implantar suas novas funcionalidades.

A tabela 4.4 mostra uma visão geral de todo o processo de desenvolvimento de software e dar uma melhor dimensão dos passos e ações seguidas durante o decorrer de todo esse processo.

Tabela 4.4: Ações de cada fase do processo

Fase	Ações
Planejamento e Modelagem	- Plano de desenvolvimento do software - Levantar requisitos - Definir funcionalidades - Gerenciamento de risco (planejamento, identificação, análise, buscar respostas) - Orçamento do projeto - Levantar recursos - Fazer um cronograma - Modelagem
Desenvolvimento	- Divisão de equipes e responsabilidades - Codificação das funções do sistema (feita por etapas) - Monitoramento e controle de riscos - Gerenciamento de novos riscos - Testar funções para descobrir falhas - Integração das funções codificadas
Testes	- Levantar o perfil operacional do sistema - Construir conjunto de dados de teste - Testar as funcionalidades do sistema - Executar processos de verificação e validação - Testar a disponibilidade - Avaliar a dependabilidade do sistema (se os mecanismos de tolerância a falha implantados funcionam) - Monitoramento e controle de riscos - Gerenciamento de novos riscos - Remover defeitos - Analisar o fluxo de dados - Fazer um relatório do comportamento do sistema - Medir a quantidade de falhas ocorridas, o tempo para ocorrer uma falha - Identificar quais os tipos de falhas que ocorreram - Atribuir uma medida de confiabilidade do sistema - Avaliar a confiabilidade do sistema
Implantação	- Implantar o sistema no ambiente de uso - Acompanhar o funcionamento - Manutenções corretivas e preventivas - Monitoramento e controle de riscos - Modificações e inclusão de recursos - Gerenciamento de novos riscos - Novos testes

5 CONCLUSÃO

A necessidade por softwares que possam ser tolerantes a falhas e que estejam disponíveis no momento que precisar ser feito uma requisição de serviço tem motivado a busca de confiabilidade e proteção de software. A confiabilidade vai garantir além da disponibilidade, uma execução de serviços com total qualidade. A proteção servirá para proteger o sistema de riscos que possam se manifestar durante seu uso. Por isso, são dois fatores indispensáveis em qualquer sistema de software.

No entanto, para poder obter a confiabilidade e proteção de software que se deseja, é preciso adotar um processo de desenvolvimento de projeto que seja confiável. Desse modo, foi estudado nesse trabalho quais as técnicas e mecanismos que precisam ser adotados e como o processo deve ser seguido. Esse processo precisa utilizar meios de análise e recuperação de falhas, englobar gerenciamento de riscos, testar a confiabilidade do sistema, e, sobretudo, possuir uma fase de planejamento de todo o projeto. Que consistirá para levantar todos os requisitos necessários, fazer o planejamento do gerenciamento de riscos, e construir modelos para o projeto que se pretende desenvolver.

A conclusão a qual se chegou foi que nem sempre é fácil escolher um processo adequado para garantir a confiabilidade do software. Desse modo, o processo proposto foi baseado em todos os requisitos de confiabilidade descritos. Acredita-se que ele seja seguro e trará a confiança esperada pelos usuários do software. No entanto, uma das grandes dificuldades dos processos de desenvolvimento de software é a certeza de que realmente o sistema atingiu a medida exata de confiabilidade que se esperava, e sem dúvida, também foi uma dificuldade do processo proposto. Outra dificuldade foi a escolha de métodos de testes que sejam eficazes para medição da confiabilidade e como deve ser feita a análise dos resultados para se estimar a confiabilidade do sistema.

Como trabalhos futuros, pretende-se fazer um estudo de qual é o impacto provocado pela gestão de risco nos negócios corporativos das empresas. E através desses dados, melhorar o gerenciamento de riscos dentro do processo proposto, como forma de aumentar cada vez mais a confiabilidade dos sistemas desenvolvidos por meio desse processo.

REFERÊNCIAS

ALMEIDA JR, Jorge Rady. **Segurança em sistemas críticos e em sistemas de informação – Um estudo comparativo**. Tese (Livre Docência). Escola Politécnica da Universidade de São Paulo. São Paulo, 2003.

ANDRADE, Michelle C. Alves. **Gerência de riscos: um processo simplificado para pequenas empresas de software**. Monografia apresentada na Universidade Federal de Lavras. Lavras, 2007.

BUENO, Cassiane F. S; CAMPELO, Gustavo B. **Qualidade de Software**. Artigo publicado na Universidade Federal de Pernambuco. Disponível em: <http://www.riopomba.ifsudestemg.edu.br/dcc/dcc/materiais/1022789570_Qualidade de Software.pdf> Acesso em: 24 de fevereiro de 2013.

FERREIRA, Enderson. **Análise de confiabilidade de sistemas redundantes de armazenamento em discos magnéticos**. Dissertação (Mestrado em Engenharia). Escola Politécnica da Universidade de São Paulo. São Paulo, 2003.

GOMEZ, Thiago Coelho. **Gerenciamento de riscos utilizando o PMBOK**. Monografia apresentada na Faculdade Lourenço Filho. Fortaleza, 2010.

LYU, M. R. **Software Reliability Engineering: A Roadmap**. Future of Software Engineering, FOSE 07. IEEE, 2007.

MUSA, J. D. **Software reliability – engineered testing**. Computer, v. 29, n. 11, p. 61-68, 1996.

SMITH, Jonathan M. **A Survey of Software Fault Tolerance Techniques**. Department of Computer Science, Columbia University. New York, 1988.

SOMMERVILLE, Ian. **Engenharia de Software**. 9 ed. São Paulo: Pearson, 2011.

SOUZA PURRI, Marcelle C. M. **Estudo e propostas iniciais para a definição de um processo de desenvolvimento para software científico**. Dissertação (Mestrado em Modelagem Matemática e Computacional). CEFET-MG. Belo Horizonte, 2006.

VASCONCELLOS NETO, O. C. **Análise de disponibilidade em sistemas de software na web**. Dissertação (Mestrado em Engenharia). Escola Politécnica da Universidade de São Paulo. São Paulo, 2009.

WEBER, T. S. **Tolerância a falhas: conceitos e exemplos**. Artigo publicado na UFRGS. Porto Alegre, 2003.

WEBER, T. S. **Um roteiro para exploração dos conceitos básicos de tolerância a falhas**. Artigo publicado na UFRGS. Porto Alegre, 2002.