EVERSON CARLOS MAUDA

MODELO DE QUALIDADE PARA CARACTERÍSTICAS INTERNAS DE SEGURANÇA DE COMPONENTES DE SOFTWARE

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná para obtenção do título de Mestre em Ciências.

Curitiba
2012

EVERSON CARLOS MAUDA

MODELO DE QUALIDADE PARA CARACTERÍSTICAS INTERNAS DE SEGURANÇA DE COMPONENTES DE SOFTWARE

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná para obtenção do título de Mestre em Ciências.

Área de concentração: Ciência da Computação

Orientadora: Profa. Dra. Sheila dos Santos Reinehr
Co-Orientador: Prof. Dr. Altair Olivo Santin

Curitiba
2012

Dados da Catalogação na Publicação
Pontifícia Universidade Católica do Paraná
Sistema Integrado de Bibliotecas – SIBI/PUCPR
Biblioteca Central

M447m 2012

Mauda, Everson Carlos
Modelo de qualidade para características internas de segurança de componentes de software / Everson Carlos Mauda ; orientadora: Sheila dos Santos Reinehr ; co-orientador: Altair Olivo Santos. – 2012.
205 f. : il. ; 30 cm

Dissertação (mestrado) – Pontifícia Universidade Católica do Paraná, Curitiba, 2012
Bibliografia: f. 144-160

1. Software - Controle de qualidade. 2. Software - Medidas de segurança. I. Reinehr, Sheila dos Santos. II. Santin, Altair Olivo. III. Pontifícia Universidade Católica do Paraná. Programa de Pós-Graduação em Informática. IV. Título.

CDD 22. ed. – 005.10685

DEDICATÓRIAS

Aos meus amados pais Gerson e Marilena.

A minha doce Vanessa.

A todos que sempre acreditaram em meu potencial.

AGRADECIMENTOS

A Trindade Santa, Deus Pai, Deus Filho e Deus Espírito Santo, por todas as graças e bênçãos deixadas no meu caminho, pela força na hora das provações e por mais essa batalha vencida com sucesso.

Aos meus pais, Gerson e Marilena, que sempre deram apoio e suporte sobre as minhas decisões e me ajudaram a dar meus primeiros passos na vida e nos valores morais, éticos e católicos, tornando-me um homem honesto e digno de apreciar a vida.

A minha esposa Vanessa que esteve sempre no meu lado nos momentos de alegria, de raiva, de tristeza, de desanimo. Foram muitas horas dedicadas a esse trabalho na frente de computadores, artigos, livros, deixando-a muitas vezes como segundo plano. Obrigado pela paciência e carinho.

Aos meus familiares que em sua maioria entenderam o meu esforço e minhas ausências e comigo celebram mais uma conquista.

A minha orientadora Sheila Reinehr pela confiança e ajuda nos momentos de desenvolvimento e revisões deste trabalho.

A professora Andreia Malucelli pela confiança, ajuda na elucidação de ideias e pelas palavras confortantes nos momentos mais sombrios. Agradeço também por proporcionar a oportunidade de evoluir o meu dom do ensino, que eu considero o dom mais gratificante que um ser humano pode possuir.

Ao professor Altair Olivo Santin por sua solicitude em me ajudar e por todas as horas ocupadas em conversas, emails e materiais para compreender a gigantesca e importante área de segurança de software.

As amigas Joselaine Valaski, Andressa Ianzen e Kelly Bettio e aos amigos João Coelho e Cleiton Garcia por suas amizades e principalmente pela alegria para enfrentar a difícil batalha do mestrado.

Aos padrinhos Gilberto Conor e Gustavo Laufer e suas respectivas patroas pela perseverança e força de nossas amizades desde a época da graduação na Universidade Federal do Paraná.

Aos amigos Antônio Quiroga, Daniel Brizuena, Guilherme Arantes, Juliano Haus, Marcelo Ferri e Wellington Moscon pela alegria da convivência no trabalho da Cinq Technologies e pelas jantas e rodadas de poker e uno que ajudavam a distrair uma mente ocupada que estava a mil por hora.

Aos amigos Alexandre Mikio, Glaucio Polzin, Ivandro Sá, José Guimarães, Marcos Paulino e Rangel Jungles, Paulo de Tarso e Paulo Pinheiro pela alegria e animação durante a convivência do trabalho na Serpro, ajudando a suportar a “pressão” do dia a dia.

A todos aqueles que confiam e esperam no meu trabalho. Obrigado.

“Terminar o momento, encontrar o final da jornada em cada passo do caminho, viver o maior número de boas horas, é sabedoria”
- Ralph Waldo Emerson

“É bom ter um fim para a jornada, mas é a jornada que importa no fim.”
- Usula K. LeGuin

“Em toda minha vida profissional, jamais acreditei em messianismo, estrelismo, concentração do poder e do mérito em um só indivíduo. Sempre trabalhei em equipe. E se algum merecimento tenho, é o de ter sabido despertar em meus companheiros o entusiasmo, delegar-lhes autoridade com responsabilidade, exortá-los ao pleno uso de suas potencialidades e qualidades.”
- Marechal Casimiro Montenegro Filho

RESUMO

Componentes de software são um meio pelo qual o reuso de software pode ser alcançado, visando melhorar a qualidade e a produtividade das equipes de desenvolvimento. Uma das principais preocupações do desenvolvimento de componentes está relacionada com a segurança da informação. Uma das formas de tornar um componente mais seguro é melhorar a segurança de suas características internas. Modelos de qualidade de software descrevem características de qualidade para sistemas baseados em software. No entanto, estes, em geral, não focam sobre os aspectos relacionados à segurança. O objetivo principal deste trabalho é criar um modelo de qualidade para componentes com foco nos itens internos de segurança, criando um guia para arquitetos e desenvolvedores de software que não possuem um vasto conhecimento sobre a área de segurança, ajudando-os nas decisões sobre a inserção de itens de segurança em seus projetos de software. O método utilizado contemplou as seguintes fases: (i) Analisar o cenário de modelos de qualidade para componentes; (ii) Analisar as características de segurança para componentes; (iii) Definir o Modelo de Qualidade para Características internas de Segurança para Componentes de Software; (iv) Avaliar o modelo. Seguindo a metodologia proposta, o modelo foi construído e submetido à avaliação de especialistas, por meio de um questionário de graduações usando a escala de *Likert* e de questões abertas para a coleta de sugestões. Após a análise das avaliações, foi realizado o refinamento do modelo de qualidade. O modelo é composto por cinco grandes áreas da segurança da informação: Criptografia de Dados, Autenticidade e Identificação, Auditabilidade, Controle de Vulnerabilidades e Manipulação de Dados. A avaliação pelos especialistas apontou para algumas melhorias que foram incorporadas ao modelo, gerando sua versão final. Cinco especialistas provenientes tanto da academia, quanto do mercado profissional, todos com experiência em segurança responderam o questionário. A geração do modelo de qualidade representa uma contribuição importante para a área, uma vez que não foi possível encontrar nenhum outro modelo que cobrisse os aspectos abordados por este trabalho. O modelo pode ser utilizado, para melhorar a questão da segurança em seus componentes e softwares.

Palavras-chaves: Componentes de Software; Segurança Interna de Componentes de Software; Modelo de Qualidade de Componentes de Software.

ABSTRACT

Software components are a means of reaching software reuse, aiming at improving development team's quality and productivity. One of the main concerns of component development is related to information security. One way to become a component more secure is to improve the security of its internal characteristics. Software quality models describe quality characteristics for software based systems. But these models, most of the times, do not focus on security specific aspects. The main goal of this research project is to propose a quality model for components with a focus on their internal security aspects. It also aimed at creating a guide for architects and software developers, that do not have a broad knowledge on security, to help them in decisions about the inclusion of secutrity items within their software projects. The research method was composed by four phases: (i) analyze the components quality models scenario; (ii) analyze security characteristics for components; (iii) Define a Quality Model for Software Component Internal Security Characteristics; (iv) assess the model. Following the proposed methodology, the model was build and sent to the specialist's assessment, using a questionnaire with Likert scale questions and open questions. After the assessment analysis, the model was refined using the suggestions received. The model is composed by five broad security information areas: Data Encryption, Access Control, auditability, Vulnerabilities Control and Data Manipulation. The assessment performed by the specialist showed some improvement suggestions that were added to the model, generating its final version. Five specialists, coming either from academia, as from the professional market, have answered the survey. All of them had previous solid experience with security issues. The generation of a quality model, represents an important contribution to the field, once it was not possible to find a model that covered all the aspects developed by this work. It was possible to present a model that can be used by software developers and architects, in order to improve the security issues in their software and components.

Keywords: Software Component Certification, Internal Security of Software Components, Software Component Quality Model.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interfaces
BSD	Burkley Software Distribution
CBD	Component-Based Development
CBSE	Component-Based Software Engineering
CC	Common Criteria
CCM	CORBA Component Model
CMM	Capability Maturity Model
CMU/SEI	Carnegie Mellon University's Software Engineering Institute
COM	Component Object Model
CORBA	Common Object Request Broker Architecture
COTS	Commercial-Off-The-Shelf
CRHF	Collision Resistant Hash function
DCOM	Distributed Component Object Model
ECC	Elliptic Curve Cryptograpy
EJB	Enterprise Java Beans
EAL	Evaluation Assurance Levels
FEK	File Encryption Key
FSK	File Signature Key
GQM	Goal Question Metric
IETF	Internet Engineering Task Force
ISO	International Organization for Standardization
ITEA	Information Technology for European Advancement
MDC	Modification Detection Codes

MSFR	Minimum Security Functionality Requirements for Multi-user Operating Systems
NIST	National Institute of Standards and Technology
OMG	Object Management Group
OSS	Open Source Software
OTS	Off-The-Shelf
OWASP	Open Web Application Security Project
OWHF	One-Way Hash Function
PIN	Personal Identification Number
PP	Protection Profile
RFC	Request for Comment
RIPE	Réseaux IP Européens
ST	Security Target
SPLE	Software Product Line Engineering
SQuaRE	Software Product Quality Requirements and Evaluation
TCSEC	Trusted Computing System Evaluation Criteria
TOE	Target Of Evalutation
WEP	Wired Equivalent Privacy
WORM	Write Once, Read Many
XML	eXtensible Markup Language

CAPÍTULO 1 - INTRODUÇÃO

A maioria dos softwares atuais são como as pirâmides do Egito, com milhões de blocos empilhados uns em cima dos outros, nenhuma integridade estrutural, feita apenas pela força bruta e milhares de escravos.
– Alan Kay – Co-criador da Linguagem de Programação Smalltalk e do paradigma de orientação a objetos

Nos últimos anos, o desenvolvimento de sistemas baseado em reúso de software tornou-se uma alternativa economicamente viável em vista do processo tradicional de desenvolvimento de software. Isso decorre da "utilização de artefatos já existentes ao invés de realizar a sua construção a partir do zero (…), reduzindo o tempo de construção e melhorando a qualidade do produto final" (KRUEGER, 1992). Infelizmente quando o reúso não é planejado corretamente, há o dispêndio de tempo e recursos dentro da organização. Um agravante maior está no fato de que se a implantação da cultura de reúso falhar, pode tornar os gerentes e superiores receosos quanto a novas tentativas para a aplicação do reúso de software (ALVARO ET AL., 2010).

O reúso possui uma amplitude grande, existindo diversas formas de realizar a sua aplicação, mas não existe uma "receita de bolo" de como aplicar o reúso, pois isso depende da cultura da empresa (FRAKES; ISODA, 1994) (ALMEIDA ET AL., 2007). Independentemente da forma de aplicação, o objetivo principal é evitar o redesenvolvimento de algum artefato, buscando soluções já prontas dentro da organização ou no mercado.

Uma dessas formas de implementar o reúso é o Desenvolvimento Baseado em Componentes (*Component-Based Development* – CBD), que utiliza componentes de software para realizar o desenvolvimento de projetos. De acordo com Capretz, Capretz e Li (2001) "componente de software é um bloco de lego". Ao realizar o encaixe desses blocos, é obtido o software.

A partir da evolução das linguagens orientadas a objetos, o componente passou a ter grande importância dentro da construção de um projeto de software,

principalmente pelos benefícios que este traz para o projeto, como, redução do tempo de entrega do projeto e dos custos de manutenção e melhoria sobre a qualidade do projeto em geral (D'SOUZA; WILLS, 1999) (VOAS, 1999).

De acordo com a Softex (2007), os componentes de software estão em aquecimento na economia mundial, indicando que o faturamento está crescendo em uma ordem de 40% ao ano, como mostra a Figura 1-1, movimentando em 2002 mais de 1 bilhão de dólares no mercado mundial. Outro dado importante é que em 2003, 70% dos novos projetos utilizaram componentes prontos ou fabricaram componentes para produzir as regras de negócio.

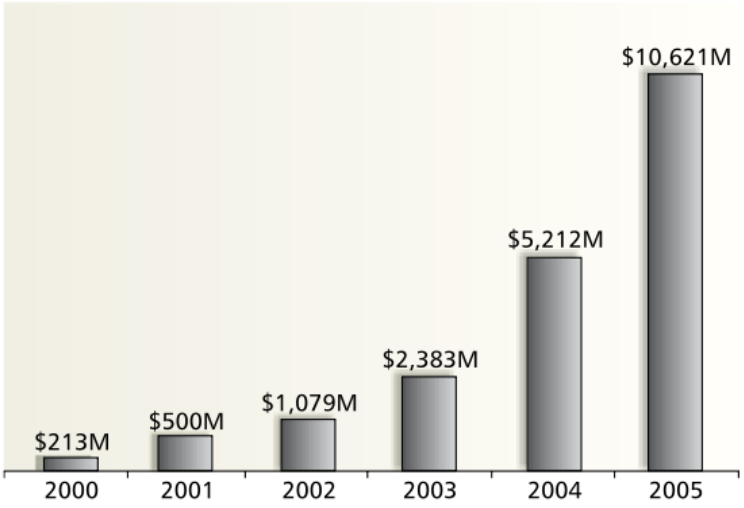

Figura 1-1. O mercado de componentes de software 2000 a 2005, adaptado de (SOFTEX, 2007).

Diante deste cenário, surge uma indagação: se os componentes são tão importantes, porque ainda não há uma ampla difusão e utilização? Um estudo feito pelo *Software Engineering Institute* da *Carnegie Mellon University* (CMU/SEI) por Bass et al. (2001), afirmou que os principais fatores que inibem a utilização dos componentes pelos desenvolvedores e organizações são:

- Falta de componentes disponíveis
- Falta de padrões estáveis para a tecnologia de componentes
- Falta de componentes certificados
- Falta de um método de engenharia para produzir consistentemente sistemas de qualidade a partir de componentes

Como se pode observar, um dos inibidores é a falta de componentes certificados, significando que existe uma indagação sobre a sua qualidade. Uma forma de promover uma melhor qualidade encontra-se no planejamento do projeto

de software. De acordo com Kalaimagal e Srinivasan (2009) a equipe de planejamento de um projeto deve estar apta para identificar danos a um sistema pela inserção de um novo componente.

Mesmo com uma equipe preparada, erros podem ocorrer, como o caso do foguete Ariane 5 (JEZEQUEL; MEYER, 1997). Em junho de 1996, o foguete Ariane 5 explodiu quarenta segundos após a decolagem. Relatórios indicaram que a explosão foi causada por um erro de software em uma conversão de ponto flutuante, excedendo o limite de bits. Como não havia nenhum manipulador de exceções, esta seguiu o destino habitual, travando e deixando sem controle o computador de bordo do foguete. O relatório principal informou que esse componente havia sido testado e utilizado durante a decolagem do Ariane 4, não apresentando nenhum tipo de falha (LIONS, 2011). Este ainda afirma que não havia diferença significativa entre os sistemas dos foguetes, assim o componente deveria ter funcionado adequadamente.

Esse exemplo identifica a existência de várias razões para dificultar uma consolidação de requisitos de qualidade em componentes, pois não há um consenso sobre quais características de qualidade deveriam ser avaliadas.

Assim existem diversos modelos de qualidade que tentam chegar a essa consolidação. As normas e a maioria dos modelos de qualidade de software são hierárquicos, isto é, definem características que agrupam subcaracterísticas. Essas subcaracterísticas definem atributos que são utilizados para avaliar o software.

Uma característica importante de alguns desses modelos é a relacionada à segurança. Essa área, que sempre foi considerada como uma estratégia de gerenciamento de risco, está sendo visualizada atualmente como uma vantagem competitiva, principalmente em áreas financeiras e em áreas comerciais (SHIN; WILLIAMS, 2008).

1.1 Motivação

A importância que os componentes de software vêm representando para o desenvolvimento de sistemas em larga escala, principalmente nas instituições que realizaram uma boa adoção do reúso no seu dia a dia foi um dos motivadores dessa pesquisa.

Ao inserir um componente em um sistema em que este não tenha suas características de segurança devidamente testadas ou especificadas, expõe a empresa a vulnerabilidades como, por exemplo, invasões de servidores, obtenção

de dados de maneira ilícita, entre outros problemas que normalmente são causados por hackers e outras pessoas com intenção de prejudicar o sistema.

Shin e Williams, (2008) definem vulnerabilidade como uma instância de uma falta na especificação, desenvolvimento ou configuração de um software, tal que sua execução pode violar uma política de segurança implícita ou explícita do sistema. Uma vulnerabilidade pode ser classificada em três classes principais (AVIZIENIS ET AL., 2004):

- **Falha**: Evento que ocorre quando a funcionalidade sofre um desvio do seu funcionamento correto. Pode estar relacionado a um problema na especificação funcional do sistema.
- **Erro**: É a causa de uma falha, ou seja, do desvio do funcionamento correto do sistema.
- **Falta**: É a declaração ou hipótese de um erro. Pode ser ativa, quando causa o erro ou inativa, quando não o causa. E pode ser interna ou externa ao sistema.

De acordo com Nist (2002), o custo estimado para falhas e faltas na segurança dos sistemas de software é de 60 bilhões de dólares por ano. Prejuízos causados por falhas de segurança podem ser facilmente identificados, por exemplo, no caso de alguns vírus como o Code Red (2001) e o Melissa (1999), que causaram prejuízos de 2,1 bilhões de dólares e 1,1 bilhões de dólares, respectivamente. Ambos se utilizaram de falhas em softwares para se infiltrar no sistema.

De acordo com Nist (2009), existem algumas abordagens ainda a serem estudadas sobre a área de métricas de segurança, como a definição de modelos formais de métricas de segurança, a análise de dados históricos, as técnicas de avaliação utilizando inteligência artificial, os métodos de medição concretos e praticáveis, além dos componentes intrinsecamente mensuráveis.

Os modelos de qualidade para componentes de software atuais não tratam as características de segurança de um componente em um nível de profundidade adequado. Dessa forma a elaboração de um modelo de qualidade que possa ajudar desenvolvedores e arquitetos a definir melhor esses pontos de um componente de software é de grande importância.

1.2 Objetivos

Como há um número muito pequeno de modelos de qualidade para componentes e ainda mais limitado é o número de modelos de qualidade para segurança de componentes, é necessário realizar uma investigação dos conceitos teóricos que podem ser aplicados em um modelo de qualidade com foco no aspecto de segurança de componentes.

Dessa forma esse trabalho visa auxiliar na melhoria da qualidade dos componentes de software, dentro dos aspectos de segurança, contribuindo para a melhoria no desenvolvimento e na qualidade dos componentes.

A partir desse propósito delimita-se o objetivo geral desse trabalho que é **propor um modelo de qualidade para características internas de segurança de componentes de software.**

Para atender ao objetivo principal da pesquisa, os seguintes objetivos específicos devem ser realizados:

(I) Identificar as grandes áreas de segurança para software e extrair as características principais para a construção do modelo;

(II) Construir um modelo de qualidade para características internas de segurança em componentes de software; e

(III) Avaliar o modelo de qualidade proposto.

A questão primordial que se pretende responder pode ser sintetizada da seguinte forma: "**É possível definir as características internas de segurança de um componente de software?**".

1.3 Delimitação de escopo

Para Cervo (2002), delimitar o escopo é selecionar um tópico ou parte a ser focalizada. E complementa: "convém superar a tendência muito comum de escolher temas que, por sua extensão e complexidade, não permitam estudos em profundidade." Portanto, faz-se necessário a delimitação do escopo desta pesquisa para que seus resultados e a sua qualidade possam ser alcançados dentro do prazo disponível.

Ao realizar a explanação das características internas de qualidade, não se pretende substituir a opinião do arquiteto de projeto de software, nem tão pouco excluir componentes que não estejam nos padrões avaliados. O modelo visa a

ajudar a verificação, por parte do arquiteto, se determinados itens de segurança de software estão presentes ou deveriam ser inseridos no projeto.

Os processos de desenvolvimento de software não estão sendo retratados nesse trabalho, mas sim a parte técnica sobre o desenvolvimento de componentes com melhores características de segurança. Isso delimita o escopo, pois não há um tratamento no modelo sobre o sigilo dos dados aos quais uma pessoa que trabalha no desenvolvimento do componente tem acesso. Assim como não há uma preocupação com a melhora de processos de reúso, versionamento e publicação de componentes. Os bens ativos de uma empresa também não são levados em consideração nesse trabalho.

Outro aspecto a ser excluído do escopo desse trabalho é a parte de integração de componentes, não integrando assim formas de autenticação únicas para vários componentes, integração entre plataformas Altas, por exemplo, Cobol, Natural e plataformas Baixas, por exemplo, Java e Asp. Isso é um aspecto relevante que é descrito para trabalhos futuros.

O modelo de qualidade não pretende ser um fim, mas apenas um meio de contribuir para um aumento de atenção dos desenvolvedores e arquitetos, pessoas ligadas ao desenvolvimento de software, para que seja dada a devida importância às características de segurança de um componente de software.

1.4 Processo de trabalho

Com o objetivo de organizar o trabalho de pesquisa a ser realizado, foi definido um processo inicial contendo fases, atividades e resultados esperados. As fases são as seguintes:

- Fase 1 – Preparação da Pesquisa: fase que corresponde à delimitação da área de estudo, coleta e análise das referências bibliográficas, delimitação do tema e estabelecimento dos objetivos, questões e proposições.
- Fase 2 – Estruturação da Pesquisa: fase de elaboração de um quadro referencial teórico, seleção do método de pesquisa e definição das etapas de pesquisa.
- Fase 3 – Execução da Pesquisa: fase da investigação em si, com busca e análise de trabalhos na literatura, desenvolvendo o modelo de qualidade, juntamente com suas características, subcaracterísticas, atributos e tipos.

- Fase 4 – Avaliação dos Resultados: fase da análise dos resultados obtidos da avaliação por especialistas extraindo as generalizações e conclusões.

1.5 Estrutura do documento da dissertação

Esse documento está organizado da seguinte forma:

- O Capítulo 1, aqui apresentado, visa oferecer ao leitor um panorama geral sobre o contexto que se insere este trabalho de pesquisa. E com esse panorama apresentado, definir o objetivo geral e os objetivos específicos, apresentando em seguida o processo de trabalho.
- O Capítulo 2 aprofunda o referencial teórico inicial descrito no Capítulo 1, focando os temas de reúso de software, componentes de software e mais especificamente, modelos de qualidade de software juntamente com os conceitos das grandes áreas de segurança de software.
- O Capítulo 3 apresenta um posicionamento metodológico, definindo a estruturação detalhada da pesquisa, com as proposições iniciais fundamentadas na investigação bibliográfica realizada no capítulo 2.
- O Capítulo 4 descreve o modelo de qualidade para características internas de segurança, apresentando os pontos chave definidos sobre cada característica, subcaracterística e atributo de segurança.
- O Capítulo 5 discute os resultados da pesquisa de avaliação do modelo de qualidade realizada com especialistas
- O Capítulo 6 conclui este trabalho, destacando a relevância deste trabalho e apresentando as considerações finais, abrindo as portas para outros trabalhos futuros.
- O Apêndice A destaca a pesquisa realizada com os especialistas.

1.6 Considerações sobre o capítulo

Este capítulo apresentou a importância dos componentes de software nos projetos desenvolvidos pelas empresas produtoras de software. Apresentou como a falta de qualidade dos componentes, principalmente no aspecto de segurança, pode deixar um projeto de software menos confiável. Isso se torna a motivação para esse trabalho que foi apresentada, juntamente com os objetivos, a delimitação do escopo e o processo de trabalho que será desenvolvido nessa dissertação.

CAPÍTULO 2 - REVISÃO DA LITERATURA

Antes do software poder ser reutilizável ele primeiro tem de ser utilizável.

– Ralph Johnson - Co-autor do livro Design Patterns: Elements of Reusable Object-Oriented Software

A indústria da informática sempre teve dois protagonistas, o hardware e o software. Até o fim dos anos 60, o hardware era o protagonista principal, pelo avanço necessário na criação de processadores, memórias e dispositivos de entrada e saída. A partir do início dos anos de 1970, este cenário mudou e o software passou a ser utilizado em grande escala nas empresas, aumentando a complexidade e necessitando de uma maior robustez para sofrer manutenções. Pressman (2001) descreve essa mudança de protagonistas na história da informática:

> "Melhorias dramáticas no desempenho do hardware, mudanças profundas na arquitetura de computadores, enormes aumentos na memória e capacidade de armazenamento e uma ampla variedade de opções exóticas de entrada e saída, tem precipitado uma maior complexidade e maior sofisticação em sistemas computacionais. Sofisticação e complexidade podem produzir resultados brilhantes quando um sistema é bem sucedido, mas podem também representar grandes problemas para aqueles que realizam a construção dos sistemas complexos." (p. 4 e 5, tradução nossa)

Brooks (1987) constata que muitos dos problemas clássicos de desenvolvimento de software derivam de um aumento de complexidade e tamanho do software. Com isso aumentam os problemas de comunicação entre as pessoas de um projeto de software, as dificuldades no entendimento das regras de negócio, as falhas, os custos excessivos e os atrasos no cronograma. Com o intuito de melhorar esse cenário, começam a surgir projetos de pesquisa para a concepção de novas técnicas e metodologias de desenvolvimento de software como descreve Sommerville (2001):

> "Uma forma organizada de produzir software inclui sugestões sobre o processo a ser seguido, notações a serem utilizadas, regras que regem as descrições do sistema que são produzidas e as orientações para o *design*. " (p. 17, tradução nossa).

Com essas pesquisas se estendendo por muitos anos, cada década pôde ser caracterizada por uma tecnologia melhor difundida no momento, sendo esta dominante durante a década e, às vezes, pós década (ALMEIDA ET AL., 2007).

Na década de 1950 um computador custava mais de dois milhões de dólares e para construir um software gastava-se em torno de um oitavo desse valor. Sendo um custo muito elevado, somente grandes organizações podiam adquirir um computador. Não havia a manutenção de software, pois ao substituir o hardware era necessário criar um novo software para funcionar neste hardware (BROOKS, 1987). Outro ponto a ser mencionado é que esses softwares estavam contidos em um único arquivo, sendo uma limitação das linguagens de programação da época.

Nessa época as linguagens de programação possuíam o conceito de *GOTO*, ou seja, saltar instruções de linhas de código, avançando ou retrocedendo sua execução. Mas a utilização de forma errônea do *GOTO* traz grandes malefícios à legibilidade do código, a localização de erros e principalmente em sua execução, pois podem ocorrer *deadlocks*. De acordo com Hildering (2003) *Deadlock* é:

> "O fracasso da sincronização entre dois ou mais processos, de forma que estes não são capazes de chegar a um acordo sobre um evento comum, embora estejam dispostos a participar em outros eventos; esse conflito de sincronização ocorre durante a execução do software." (p. 289, tradução nossa).

Em um de seus artigos mais célebres Dijkstra (1968) menciona seu desapontamento com o conceito de GOTO:

> "Muito recentemente eu descobri o porquê do uso do GOTO pode gerar efeitos desastrosos, e eu me convenci que o GOTO deveria ser abolido de todas as linguagens de programação de "Alto Nível", isto é, todas exceto, talvez, a linguagem de máquina." (p.147, tradução nossa).

A década de 1960 foi caracterizada pela *técnica estruturada*, realizando a quebra do paradigma do GOTO. Um programa é do tipo estruturado, se dentro dele não há códigos GOTO de um segmento de código para outro segmento de código (ALMEIDA ET AL., 2007). Um dos principais conceitos da técnica estruturada é o de sub-rotinas, isto é, na quebra de um código monolítico com várias responsabilidades e tarefas, em pequenos pedaços de código mais especializados e com uma única responsabilidade, resolvendo somente um pequeno problema.

Assim, o *GOTO* deixou de existir, sendo substituído por *chamadas de sub-rotinas*. Cada sub-rotina pode chamar ou ser chamada por outras, estruturando um código para resolver um problema maior. O uso da técnica estruturada se difundiu

nas universidades e organizações, mas muitos softwares eram concebidos individualmente e não para várias aplicações. Pressman (2001) descreve da seguinte forma:

> "Na década de 1960, nós construímos muitas bibliotecas com sub-rotinas científicas, as quais foram utilizadas em uma grande gama de aplicações de engenharia e científicas. Essas bibliotecas implementavam bons algoritmos de uma forma muito eficaz, mas tinham um domínio de aplicabilidade limitado." (p. 9, tradução nossa)

Nesse período surgiu o conceito de *Reúso de Software*, ou seja, reutilizar as sub-rotinas criadas. Isso será discutido com mais detalhes no capitulo 2.1. Dessa ideia nasceu a *técnica de módulos* na década de 1970, como descreve Almeida et al. (2007):

> "Um programa é modular, se este é dividido em segmentos de códigos hierárquicos, que cada um tem entradas e saídas únicas. Esses segmentos devem corresponder a operações elementares do programa e cada operação deve estar disponível para ser invocada por rotinas de níveis mais altos." (p. 117, tradução nossa)

Muitas dessas operações a que se refere Almeida et al. são complexas, mas comuns à maioria dos programas como, por exemplo, Entrada e Saída de Informações, Alocação de Memória ou Armazenamento em Banco de dados. Com o conceito de reúso de software, os módulos são embutidos dentro das linguagens de programação, se tornando a principal vantagem, pois um desenvolvedor passa a ter o foco de atuação somente na construção e resolução dos problemas propostos, aumentando a produtividade. Algumas dessas linguagens de programação são utilizadas até hoje, principalmente em instituições financeiras, como levantou Reinehr (2008).

Na década de 1980, o software se popularizou ainda mais, criando a necessidade para diversos segmentos de aplicação. A maior parte do software necessitava de maior robustez, reusabilidade e consistência nas informações, surgindo assim a técnica de *orientação a objetos*. Essa é amplamente utilizada nos dias atuais, com as linguagens Java e C#. Pressman (2001) faz essa descrição:

> "Nós vivemos em um mundo de objetos. Esses objetos existem na natureza, nas entidades humanas, nos negócios e nos produtos que nós usamos. Eles podem ser categorizados, descritos, organizados, combinados, manipulados e criados. Portanto, não é surpresa que uma visualização orientada a objeto seria proposta para a criação de softwares – uma abstração que nos permite modelar o mundo de forma a ajudar a melhor compreender e navegar por ele." (p. 541, tradução nossa)

Desde a década de 1960 o conceito de *Componentes de Software* vem sendo estudado, por diversos pesquisadores (MCILROY, 1968), mas só começou a se tornar real na década de 1990, com a popularização da orientação a objetos. Maiores detalhes sobre a tecnologia de Componentes serão descritos na seção 2.2. A descrição da trajetória do desenvolvimento de Software é realizada por Almeida et al. (2007), citando suas principais etapas:

> "Desde as sub-rotinas nos anos 1960, passando pelos módulos nos anos 1970, objetos nos anos de 1980 e componentes nos anos 1990, o desenvolvimento de software tem sido uma história de ascensão espiral contínua de capacidade e economia crescentes, enfrentando a competitividade crescente." (p. 18, tradução nossa)

Um dos principais aspectos dessa evolução encontra-se na qualidade dos projetos gerados ao longo das décadas. Mas mesmo com essa evolução, a qualidade de um software ainda é um problema atual como descreve Meyer (2003):

> "Se eu sou um gerente de projeto, o ideal para mim é construir um produto bom. Na realidade, por desperdiçar mais dinheiro e tempo para fazer o meu produto melhor que o estritamente aceito pelo mercado, estou colocando em risco o meu produto, minha empresa e minha própria carreira." (p. 03, tradução nossa)

O investimento em qualidade de um produto está intimamente ligado à necessidade do cliente. O reúso de software pode representar o acréscimo de qualidade necessária, com um custo menor que o desenvolvimento a partir do zero.

2.1 Reúso de Software

A primeira menção ao Reúso de Software data de 1968 com o artigo convidado "*Mass Produced Software Components*" de McIlroy (1968), que foi apresentado durante a *NATO Software Enginnering Conference*. Esse artigo se tornou o marco da área de Reúso de Software, pois:

> "Não há uma Indústria do Software, mas sim uma abordagem artesanal, sem produção de componentes, mas com peças únicas de código. Ao realizar a produção desses componentes, esses não deveriam estar em qualquer lugar, mas sim classificados por diversos critérios, como precisão e desempenho." (p. 1, tradução nossa)

Durante essa conferência nasceu também a expressão "Crise do Software", que expressa a preocupação sobre a falta de qualidade na construção e manutenção de grandes sistemas de software sob um custo economicamente factível, como demonstra Ezran, Morisio e Tully (2002):

> "Nós temos nos acostumado com projetos de software que são incontroláveis e monstros horrendos, aparentemente devorando recursos infinitos e entregando produtos mal estruturados, acima do esperado e infestado de bugs e que muitas vezes falham em atender nossas reais necessidades. Sob o nome de "Crise do Software" nós estivemos aceitando esses itens nesse estado por mais de trinta anos." (p. 1, tradução nossa)

Essas primeiras citações surgiram na mesma época dos estudos relacionados sobre a técnica de módulos, mas somente com a popularização da orientação a objetos é que o reúso passou a ser mais viável, pois uma das características da técnica é a promoção do reúso através do encapsulamento, sobrecarga e herança (DETIENNE. 1991).

O reúso é definido como "o processo de criação de software utilizando artefatos de software já existentes ao invés de realizar a sua construção a partir do zero" (KRUEGER, 1992). Complementando essa definição de reúso, Frakes e Isoda (1994) definem que reúso "é o uso do conhecimento e de artefatos de engenharia de um sistema existente, para a construção de um novo sistema – é uma tecnologia para melhorar a qualidade e manutenibilidade de um software".

A primeira impressão que o reúso causa é a de somente ser aplicado ao código fonte de um software, ou seja, na inserção de blocos de código, pois "estes são fundamentais para atingir o objetivo de não reinventar a roda, e de fazer progresso com base na experiência anterior sobre o que funciona" (EZRAN; MORISIO; TULLY, 2002). Mas na verdade Selby (2005) descreve o reúso de forma mais abrangente:

> "O desenvolvimento baseado em reúso enfatiza estratégias, técnicas e princípios que habilitam os desenvolvedores a criar novos sistemas de forma efetiva, usando arquiteturas e componentes desenvolvidos previamente." (p. 495, tradução nossa)

O reúso pode ser utilizado para a exploração de similaridades entre *requisitos de projetos* (problemas e objetivos) e a *arquitetura de projetos (*categorias gerais ou modelos de solução). Adotar uma arquitetura específica pode, em algumas circunstâncias específicas, ser um requisito de alto nível. (...) Em outras palavras, requisitos e arquiteturas podem não ser independentes entre si (EZRAN; MORISIO; TULLY, 2002).

Além disso, outros itens que podem ser reutilizados são: estruturas de *design*, estruturas de implementação em nível de módulos, especificações, documentos, transformações e itens relacionados com o projeto de software como modelos de qualidade, modelos de produtividade, métricas de levantamento de custos,

processos de modelagem (FREEMAN, 1983). Para finalizar essa ideia Basili e Rombach (1991) descrevem que “todas as experiências em um processo de construção de software devem ser reutilizadas”.

Existem duas formas de aplicar o reúso dentro de uma organização, o *reúso não sistemático* e *reúso sistemático* (EZRAN; MORISIO; TULLY, 2002). O primeiro está relacionado ao conhecimento individual e à sua iniciativa em realizar o reúso, não estando relacionado conscientemente com a organização e não tendo um planejamento proativo para sua implementação. Já o reúso sistemático é uma consciência contínua em busca de oportunidades de reúso dentro de todos os produtos de trabalho.

Para que se possa perceber o ganho efetivamente do reúso, devem ser criados guias, processos e métricas sendo estes coletados para análise de desempenho (PRIETO-DIAZ, 1993). Mesmo que organização não tenha um programa de reúso sistemático, pode ter bons resultados se for madura e bem administrada, Frakes e Isoda (1994) indicam que:

> “organizações que adotam o reúso de forma a entender melhor os seus domínios têm uma enorme vantagem competitiva (...), pois se uma organização não adota Reúso de Software antes de seus competidores, esta provavelmente estará fora do mercado e com grande possibilidade de falência. (...) implementar um programa de reúso com sucesso é difícil e arriscado. Pois não há um livro de receitas, cada organização deve analisar suas necessidades e objetivos. Não realizando isso pode ocorrer uma falha na adoção do Reúso que custará precioso tempo e recursos, tornando os gerentes céticos, não dispostos a arriscar novamente”. (p 15, 17 e 19, tradução nossa)

Em (EZRAN; MORISIO; TULLY, 2002) os autores enumeram alguns passos para implantar um reúso sistemático. Esses consistem na divisão do reúso de arquiteturas de projetos em três dimensões de modelos de solução. *Arquitetura funcional*, relacionada aos domínios de aplicação; *Interface com um Usuário*, relacionada à combinação de itens de interação com o usuário de uma aplicação; e, *Arquitetura de Implementação,* que define padrões de infraestrutura que a aplicação utiliza.

A Arquitetura Funcional define dois tipos de reúso, o *Reúso Vertical* e o *Reúso Horizontal* (PRIETO-DIAZ, 1993). Reúso vertical é o reúso realizado dentro de um mesmo domínio ou área de aplicação, cujo objetivo é gerar novos modelos para uso na criação de sistemas sob uma mesma área de aplicação. O reúso

horizontal é o reúso de partes genéricas em diferentes domínios ou áreas de aplicação, contribuindo para o reúso entre diferentes tipos de aplicações.

O reúso horizontal pode atuar na exploração das similaridades funcionais entre diferentes tipos de domínios ou a exploração das similaridades entre domínios técnicos, isto é, a exploração de similaridades nas outras duas dimensões de arquitetura de projetos, *Interface com o Usuário* e a *Arquitetura de Implementação*, sendo estas independentes do domínio de aplicação. Ezran, Morisio e Tully (2002) finalizam essa classificação com o seguinte comparativo:

> "Oportunidades para reúso são como defeitos: quanto mais cedo eles forem encontrados, melhor. Se um defeito é introduzido durante a fase de definição de requisitos, e for encontrado durante essa fase, o custo para a correção é pequeno; se somente for encontrado, por exemplo, na fase de testes, o custo para realizar a correção será enorme. Da mesma forma, se similaridades entre aplicações forem identificadas na fase onde os requisitos e a arquitetura está sendo determinados, e se isso leva a um reconhecimento de oportunidades para reúso, o potencial benéfico é muito maior do que se a oportunidade fosse reconhecida, por exemplo, na fase de desenvolvimento." (p. 9, tradução nossa)

Outra maneira de implantar o reúso sistemático está nos conceitos de *reúso por composição* e *reúso por geração*. Sametinger (1997) caracteriza o *reúso por composição* na utilização de componentes, mas sem modificá-los, construindo assim componentes de um maior nível a partir da combinação de componentes de menor nível. Pfleeger (2001) complementa indicando que esse reúso é *bottom-up*, onde é necessário possuir uma boa biblioteca de componentes, com buscas bem padronizadas facilitando a obtenção dos melhores componentes. Outro ponto é a criação de padrões de comunicação entre os componentes. Um exemplo é o mecanismo chamado *Pipe* do sistema Unix. Este utiliza a saída de dados de um componente como a entrada de dados de outro componente (KERNIGHAN, 1984).

Pfleeger (2001) define *reúso por geração* como a concepção de componentes para tratar as necessidades especificas de um domínio de aplicação e depois serem utilizadas em outros softwares com domínios semelhantes. Sametinger (1997) complementa isso indicando que as partes a serem reutilizadas são incorporadas junto ao software gerando padrões para reúso. O principal foco deste reúso é a busca pela representação correta de um domínio de aplicação dentro de linguagens de especificação, processos de software e metageradores (PRIETO-DIAZ, 1993).

Durante a definição de artefatos para reúso, uma forma de classificá-los é de acordo com a visibilidade de seu conteúdo. Ezran, Morisio e Tully (2002) definem quatro formas: caixa preta, caixa branca, caixa cinza e caixa de vidro.

Prieto-Diaz (1993) e Sametinger (1997) definem a forma *caixa preta* como o reúso de artefatos sem observar, conhecer ou modificar o seu conteúdo. Um exemplo são as *APIs*, ou *Application Programming Interfaces*. Estas definem as interfaces e especificações, mas não revelam a implementação (SZYPERSKI, 2002). Uma vantagem deste reúso é a validação e certificação do artefato. Szyperski (2002) ainda define a interface como sendo um conjunto de assinaturas de operações que podem ser invocadas por um cliente. Cada operação tem sua semântica especificada e serve para que o desenvolvedor do componente programe essa interface e o cliente utilize essa interface.

A forma *caixa branca* é o reúso de artefatos com o poder de realizar a sua adaptação e modificação (PRIETO-DIAZ, 1993). Essa modificação pode ser feita por questões de desempenho, encapsulamento ou para obter o conhecimento da estrutura interna (SZYPERSKI, 2002). Um ponto a ser considerado é o grau de modificação e o impacto sobre os requisitos e complexidades que o artefato possui (BASILI; ROMBACH, 1991). Outro ponto está relacionado a testes e manutenção, pois é necessário um esforço adicional para validar corretamente essa nova adaptação (SAMETINGER, 1997).

A forma *Caixa Cinza* é definida como o artefato que revela apenas uma pequena parte de sua implementação (SZYPERSKI, 2002). Essa parte consiste em uma interface de parâmetros que possuem o poder de modificar o comportamento do artefato (EZRAN; MORISIO; TULLY, 2002). Não há nenhuma outra forma de visualização ou modificação do conteúdo do artefato.

A última forma é a Caixa de Vidro, cuja característica é a necessidade de “olhar para dentro” do componente, sem realizar alterações em seu conteúdo, descobrindo as propriedades internas principalmente quando a descrição do componente é inadequada (EZRAN; MORISIO; TULLY, 2002). Um dos efeitos negativos dessa visualização do conteúdo do artefato é a criação de dependências nos detalhes de implementação, tornando fatal uma alteração nesses detalhes em uma nova versão do artefato. Outro ponto negativo está na falha da documentação sendo inexistente ou insuficiente para suprir o desenvolvedor (SAMETINGER, 1997).

Outra forma de classificação está relacionada com a procedência do artefato. Ravichandran e Rothenberger (2003) definem duas formas, a primeira está na produção do artefato internamente dentro da organização, chamado de *in-house*. A segunda forma é a aquisição do artefato no mercado. Sametinger (1997) defende que comprar artefatos no mercado deve ser levado em consideração. Mesmo que o custo inicial seja alto, pode se transformar no longo prazo, sendo mais efetivo se o artefato for bem documentado, generalizado e de alta qualidade.

Além dessas classificações, alguns autores destacam certas características da área de reúso para a melhoria do processo de reúso. Basili e Rombach (1991) descrevem que a organização define sobre uma análise se o reúso é apropriado:

> "A decisão de reutilizar experiências existentes, bem como de que forma e quando reusar, deve ser baseada na análise de retorno. O retorno do reúso não é fácil de avaliar. Nós precisamos entender os requisitos; como melhor avaliar os candidatos para preencher os requisitos; e os mecanismos disponíveis para as modificações necessárias." (p. 305, tradução nossa)

Lim (1994) realizou um trabalho de análise de dois programas de reúso de software dentro da Hewlett-Packard (HP), obtendo os seguintes resultados:

- Reúso melhora a qualidade final de um produto, pois ao reutilizar várias vezes, a quantidade de defeitos corrigidos em cada iteração acumula.
- Reúso incentiva a prevenção e remoção de defeitos dentro do ciclo de vida de um projeto, amortizando custos ao reutilizar o produto em diversos pontos. Manutenibilidade e a confiança do produto são melhoradas.
- Reúso melhora a produtividade, evitando que artefatos similares sejam produzidos e tornando pequena a criação de artefatos novos. Ou seja, reduz o tempo e recursos utilizados para desenvolver um software. No caso dos dois projetos de reúso da HP a redução no tempo foi de 42%.
- Reúso implica em redução do custo de treinamentos, facilitando a transferência de conhecimento entre projetos, focando os treinamentos em novas partes dos sistemas, ou até mesmo, em novos sistemas.
- Reúso realiza a prototipação de projetos de uma forma mais rápida, utilizando de artefatos prontos, agilizando o levantamento de requisitos com o cliente, permitindo um foco maior no domínio dos problemas, disseminando esse conhecimento por toda a organização.

A implantação de reúso de software dentro de uma organização pode ser algo muito benéfico, mas demanda uma boa aplicação de tempo e recursos para ser bem

sucedida. Existem modelos e técnicas que auxiliam a promoção do reúso como, por exemplo, a técnica de Desenvolvimento Baseado em Componentes, a qual será melhor detalhada na próxima seção.

2.2 Desenvolvimento Baseado em Componentes

McIlroy (1968) constata que a indústria do software não estava industrializada, sendo um trabalho ainda artesanal a produção de um software e uma das formas de inicializar essa industrialização seria a adoção de componentes, que deveriam estar organizados para facilitar sua busca e obtenção. Contudo demorou décadas para um consenso geral, como afirma (BASS ET AL., 2001):

> "Existem muitas razões para essa dissonância conceitual, definição do termo componente, sendo uma delas a necessidade dos fornecedores de tecnologia (e pesquisadores de software?) de diferenciar seus produtos de software, ajudando a compor a própria natureza genérica do termo componente. Independente da causa dessa dissonância é necessário definirmos nossa definição" (p. 2, tradução nossa)

Capretz, Capretz e Li (2001) definem componentes como os blocos de lego da Engenharia de Software. Isso sintetiza todo o conceito do termo componente. Uma definição mais formal foi feita por Brown e Wallnau (1998) que destaca as seguintes definições do termo *Componente*, dadas por várias partes da indústria:

- Um componente é uma parte não trivial, quase independente e substituível de um sistema, cumprindo uma clara função no contexto de uma arquitetura bem definida. Este deve estar em conformidade e prever fisicamente uma serie de interfaces. (Philippe Krutchen, Rational Software)
- Um componente em tempo de execução é um pacote dinamicamente vinculado de um ou mais programas como uma unidade, com interfaces documentadas que podem ser descobertas em tempo de execução (Gartner Group)
- Um componente é uma unidade de composição que especifica contratualmente interfaces documentadas e somente com dependências explícitas de contexto. Pode ser entregue de forma independente e está sujeito à composição por terceira parte. (SZYPERSKI, 2002)
- Um componente de negócio representa a implementação de um conceito de negócio "autônomo" ou um processo de negócio. Consiste dos artefatos de software necessários para expressar, implementar e implantar

> o conceito como um elemento reutilizável de um sistema maior de negócios. (Wojtek Kozaczynski, System Software Associates R&D Labs)

Essa quantidade de definições indica que componentes tem em sua origem diferentes formas e granularidades, demonstrando que as visões sobre o termo variam conforme os participantes do processo de desenvolvimento, do processo de implantação e do processo de manutenção de um software. Componentes podem ser divididos de acordo com sua especialidade, (HEINEMAN; COUNCILL, 2001):

- *Componentes de Interface com o Usuário*, o mais comum no mercado. Engloba todos os botões, caixas de texto, entre outras interfaces.
- *Componentes de Serviço* providenciam serviços comuns aos sistemas, como acesso a banco de dados. Um ponto importante é que todos estes utilizam uma infraestrutura adicional para realizar suas funções.
- *Componentes de Domínio* são os componentes de negócio, sendo difícil sua especificação e construção. Possuem suas próprias dependências de contexto e requerem um alto nível de conhecimento do negócio.

Um dos primeiros trabalhos que expandem a visão do Desenvolvimento Baseado em Componentes (*Component-Based Development* - CBD) é o de (PRIETO-DIAZ; FREEMAN, 1987), onde os autores propõem uma abordagem de classificação central de componentes, com boas ferramentas de busca. Essa classificação deve estar relacionada com (1) Função realizada, (2) Desempenho e (3) Detalhes da implementação. Essa nova visão começa a romper com o paradigma de blocos de código monolíticos sendo substituídos por componentes interoperáveis, diminuindo a complexidade e os custos do desenvolvimento de um novo software (SAMETINGER, 1997).

Bass et al. (2001) define CBD como os passos técnicos de design e implementação necessários para desenvolver e montar componentes na forma de sistemas e entregar os sistemas resultantes. Meyer (2003) complementa que a CBD é a fórmula 1 da engenharia de software. Uma técnica que ajudou a alavancar a popularidade do CBD é a da orientação a objetos de acordo com Capretz, Capretz e Li (2001):

> “O recurso desse modelo de desenvolvimento de software está na ênfase da reusabilidade durante a criação, e na produção de componentes reutilizáveis que podem ser úteis em projetos futuros. Isso é naturalmente suportado pelo paradigma de orientação a objetos como a herança e o encapsulamento.” (p. 1, tradução nossa)

Grande parte dos padrões do CBD foi estabelecida por grandes grupos de pesquisa ou grandes empresas consolidadas. Um dos primeiros padrões foi o *Common Object Request Broker Architecture* (CORBA) que possui o padrão CORBA *Component Model* (CCM) (OMG CCM, 2002), introduzindo o padrão de componentes para dentro de uma arquitetura distribuída. Outros exemplos são o JavaBeans e o Enterprise Java Beans (EJB) (DEMICHIEL, 2007), o padrão Component Object Model (COM), com as tecnologias COM+, Distributed COM (DCOM) (MICROSOFT COM, 2011), e o Object Management Group (OMG) (OMG, 2009).

Porém Brown e Wallnau (1998) constatam que a orientação a objetos foi um ponto inicial útil e conveniente, mas não expressa todo o leque de abstrações necessárias e que a CBD pode ser realizada sem utilizar a orientação a objetos.

Outro fator que ajudou a dar notoriedade ao CBD está no desenvolvimento da internet (D'SOUZA; WILLS, 1999). Isto trouxe um aprimoramento da tecnologia de computação distribuída e a mudança da arquitetura de projetos baseada em mainframes para uma mais leve e distribuída de cliente/servidor. Isto foi possível pelas novas linguagens de programação que foram sendo desenvolvidas.

Voas (1999) descreve algumas vantagens com relação aos componentes: *ganho instantâneo de Produtividade,* pois não é necessário escrever centenas de linhas de código para alguma situação; *Tempo de mercado*, pois ajudam a entregar projetos dentro do prazo; *Custo,* pois diminuem o custo de criação; *Mudança de filosofia*, pois um software deveria ser construído da mesma forma que o hardware. Engenheiros elétricos constroem sistemas a partir dos componentes disponíveis em catálogos. Por que não fazemos isso para o software? Finaliza o autor.

CBD pode também por fim ao antigo problema de ciclos de atualizações em massa (SZYPERSKI, 2002), pois soluções integradas requerem periódicas atualizações. Ao usar componentes, a revolução dá lugar à evolução. Atualizações de componentes individuais necessárias são realizadas "fora de fase" suavizando as atualizações. Obviamente, isso requer uma forma diferente de gerenciar serviços, mas o ganho potencial é imenso. Voas (1999) descreve que quanto mais avançar para a CBD, mais a qualidade se torna mais importante do que a manutenibilidade:

> "Eu acredito que nós iremos nos mover através do mesmo paradigma de manutenção que utilizamos com os computadores pessoais: comprar um novo computador a cada dois anos ao invés de realizar uma atualização nos componentes, como, por exemplo, o processador. Mas porque iríamos fazer

> isso? A resposta é simples: computadores pessoais já não são tão atualizáveis e são baratos o suficiente para não considerarmos os gastos a cada dois anos como razoáveis. A falta de manutenibilidade irá nos levar a pensar sobre criar mais sistemas de informação descartáveis como a solução para a manutenção." (p. 4, tradução nossa)

Componentes podem utilizar os quatro tipos de visualizações citadas no capitulo de reúso. O mais comum é a caixa preta, os quais são chamados OTS (*Off-The-Shelf*). Estes são subdivididos em componentes comerciais de catálogos chamados COTS (*Commercial-Off-The-Shelf*), e os produzidos pela comunidade de software, geralmente livre chamados OSS (*Open Source Software*) (LI ET AL., 2009). Os componentes caixa branca são normalmente desenvolvidos internamente.

Com todo esse conhecimento proporcionado pelo CBD uma nova área dentro da engenharia de software surgiu a Engenharia de Software Baseada em Componentes, CBSE. Sua definição é as práticas necessárias para executar o CBD de forma repetível para construir sistemas com propriedades previsíveis (BASS ET AL., 2001). Brown e Short (1997) descrevem algumas atividades específicas:

- **Reunião dos componentes:** Identificação de possíveis componentes para um software. Várias fontes são pesquisadas e, portanto, devem ser investigadas as propriedades e qualidades de cada componente. Não é necessário um conhecimento grande da funcionalidade que o componente irá desempenhar. Identificar o maior número de interfaces da iteração componente/sistema e verificar o comportamento são objetivos importantes, principalmente porque identifica erros em um estágio inicial.
- **Qualificar Componentes:** Após a descoberta das interfaces de um componente é necessário avaliar se estes são adequados para a inserção no software. A avaliação garante que o componente irá trabalhar como esperado se integrando a arquitetura do sistema com confiabilidade e usabilidade satisfatórias. Para a avaliação devem ser estudadas as documentações do componente ou conduzir testes de avaliação. É importante a discussão com os vendedores e usuários do componente.
- **Adaptação de Componentes**: Eliminar conflitos entre a arquitetura do sistema e dos componentes. Para contornar esse problema, pode utilizar a técnica de empacotamento de componentes, onde ações diferentes serão tomadas de acordo com a visibilidade do componente. Como por exemplo, se o componente for caixa branca deve-se alterar o código. Se for caixa

cinza deve-se utilizar uma linguagem de extensão para remover ou mascarar os conflitos. Se for caixa preta deve-se utilizar um *pattern facade* para encapsular o componente do sistema.

- **Composição de Componentes**: Consiste em combinar componentes qualificados, adaptados e construídos, compondo uma arquitetura comum estabelecida para uma aplicação.
- **Atualização de Componentes**: Consiste em atualizar os componentes para novas versões ou por outros componentes com a mesma função. Essa atividade deve ser essencial pelas definições de interfaces e uma iteração controlada entre componentes.

Com essas atividades definidas pela CBSE e os fatores positivos tornaram o componente muito benéfico para o desenvolvimento de projetos. Mas surge um problema com relação à qualidade destes. Para uma verificação sobre essa qualidade é necessário um planejamento dentro do projeto. Mas não existe um consenso sobre quais características são mais relevantes para medir esta qualidade.

2.3 Normas para modelos de Qualidade

Com a concorrência acirrada nos dias atuais as empresas buscam melhorar seus produtos de forma que tenham uma qualidade maior. Apesar de existirem diversos conceitos para a definição de qualidade, um dos mais completos foi estabelecido por Garvin (1984). A sua definição base de qualidade é um conceito complexo e de múltiplas faces a qual ele descreve em cinco diferentes perspectivas:

- **Visão Transcendental**: Chamada também de visão etérea, é algo trabalhado como um ideal, mas nunca é implementado completamente. É muito parecido com o conceito de ideal de Platão ou conceito de forma de Aristóteles. No software essa visão é o reconhecimento do público.
- **Visão do Usuário**: É uma visão concreta, baseada nas funcionalidades que irão atendê-lo. É feita uma avaliação dos produtos em um conceito de funcionalidade podendo ser muito personalizada. A confiabilidade, desempenho e usabilidade são avaliadas de forma operacional.
- **Visão do Produtor**: Tem o foco na produção do produto e após a entrega. É avaliado se o produto foi feito corretamente, evitando retrabalho. Essa visão não está relacionada somente com o produto, mas pode estar com o processo.

- **Visão do Produto**: Observa as características inerentes ao produto, sendo adotada pelos defensores das métricas. Medir e controlar as propriedades internas resulta em melhores propriedades externas. Oferece uma visão objetiva e independente do contexto de qualidade.
- **Visão Baseada em Valores**: É a forma de visão de cada participante de um projeto de software, pois estes têm diferentes pontos de vista, de acordo com sua responsabilidade. Se essa diferença não é explícita, podem surgir mal entendidos sobre a qualidade tornando a aceitação mais difícil do produto.

Para avaliar a qualidade dos softwares diversos modelos de características de qualidade têm sido desenvolvidos realizando a descrição destas. Normalmente a organização desses modelos é realizada de forma hierárquica. O padrão ISO/IEC 9126 (ISO/IEC, 2001) define característica de qualidade como sendo é um conjunto de propriedades de um produto de software, pelas quais a qualidade pode ser descrita e avaliada. A característica pode ser refinada em múltiplos níveis de subcaracterísticas.

As subcaracterísticas contêm as métricas e as propriedades mensuráveis que serão avaliadas pelo modelo de qualidade. As agregações de subcaracterísticas similares formam o nível superior da hierarquia. Normalmente, as características contidas no nível superior não possuem métricas para avaliação. Por definição são somente macro características. De acordo com Nist (2003) métricas são:

> "Ferramentas criadas para facilitar a tomada de decisão e melhorar a responsabilidade e o desempenho, através da coleta, análise e comunicação de desempenho de um grupo relevante de dados relacionados." (p. vii, tradução nossa)

Mas o que garante que as métricas a serem utilizadas são boas? Jaquith (2007) descreve algumas características que uma boa métrica deve possuir:

- Deve poder ser consistentemente medida, sem critérios subjetivos;
- Deve possuir uma coleta de dados barata, ou de forma automática;
- Deve ser expressa como uma percentagem ou número cardinal, e não com rótulos qualitativos como, por exemplo, "alto", "médio" e "baixo";
- Deve ser expressa com pelo menos uma unidade de medida como, por exemplo, "defeitos", "horas" ou "dólares"

- Deve ser específica contextualmente, ou seja, relevante o suficiente para auxiliar os atores que realizam as tomadas de decisões.

A *International Organization for Standardization* (ISO), preocupada em criar normas para permitir a correta avaliação de qualidade do produto de software, desenvolveu a Norma Internacional ISO/IEC 9126 (ISO/IEC, 2001), apresentando um modelo de qualidade de propósito geral para produtos de software. Essa norma deu origem aos primeiros modelos de qualidade voltados para componentes de software. Atualmente, a ISO/IEC 9126 está sendo substituída pela família de normas da ISO/IEC 25010 (ISO/IEC, 2005). Essas deram origem aos modelos de qualidade mais recentes e que servirão de base para esse trabalho.

2.3.1 ISO/IEC 9126

A norma ISO/IEC 9126 (ISO/IEC, 2001) teve seu inicio em 1991 com a definição de características para a avaliação de um software como um produto, pelo Subcomitê de Software (SC7) do Comitê Técnico Conjunto (JTC1). Como o escopo de avaliação de um software cresceu rapidamente, no ano 2000 foi realizada uma nova versão da norma, adicionando alguns itens que faltavam na versão de 1991.

Uma das primeiras atividades realizadas na nova norma consistiu na divisão da ISO/IEC 9126 em duas. A primeira norma continuou com o mesmo número, 9126, e trata da qualidade do produto de software. A segunda norma é nomeada com o código de 14598 e trata da avaliação do produto de software. A versão 2000 da ISO/IEC 9126 é dividida em quatro partes:

- ISO/IEC 9126-1: Modelo de Qualidade
- ISO/IEC 9126-2: Métricas Externas
- ISO/IEC 9126-3: Métricas Internas
- ISO/IEC 9126-4: Métricas da Qualidade em Uso

O modelo de qualidade definido dentro da ISO/IEC 9126 é um modelo hierárquico e composto por seis características principais de qualidade. Essas características foram levantadas a partir dos seguintes requisitos:

- Cobrir conjuntamente todos os aspectos de qualidade de software resultante da definição da ISO.
- Descrever a qualidade do produto de software com o mínimo de sobreposição.
- Conceituar o mais próximo possível da terminologia estabelecida.

- Formar um conjunto de seis a oito características por razões de clareza e manipulação.
- Identificar as áreas de atributos de produtos de software para melhor aperfeiçoamento.

A definição de cada uma das características é apresentada na norma, conforme mostram as seguintes definições:

- **Funcionalidade:** Habilidade do software de fornecer funções para atender as necessidades conhecidas e/ou implícitas quando o software é utilizado em determinadas condições.
- **Confiabilidade:** Habilidade de manter um determinado nível de desempenho ao longo do tempo quando usado em determinadas condições. Como não há um envelhecimento do software, os defeitos que ocorrem durante o processo de desenvolvimento, podem resultar em falhas no software no momento de seu uso.
- **Usabilidade:** Habilidade para o entendimento, aprendizagem e uso de um software pelo usuário, sendo este atrativo em determinadas condições.
- **Eficiência:** Habilidade de prover um desempenho apropriado, relativa ao total de recursos e tempo utilizados, sob determinadas condições.
- **Manutenibilidade:** Habilidade do software de ser modificado, o que inclui correções, melhoramentos ou adaptações para mudanças no ambiente, requisitos e especificações funcionais.
- **Portabilidade:** Habilidade de utilizar um software em diversas plataformas com pouco esforço de adaptação.

Cada uma das características está dividida em subcaracterísticas, conforme Quadro 2-1. A ISO/IEC 9126 não define métricas para as características, pois as métricas dependem do negócio e das necessidades do avaliador.

Para utilizar o modelo de qualidade, devem ser definidos os atributos de qualidade e suas métricas, pois apenas com as características e subcaracterísticas não é possível mensurar. Um ponto a ser sempre visualizado nos modelos, visando este trabalho, está nos itens de segurança. A norma 9126 define apenas uma subcaracterística chamada Segurança.

Quadro 2-1. Características e Subcaracterísticas da ISO/IEC 9126.

CARACTERÍSTICA	SUBCARACTERÍSTICA
Funcionalidade	Adequação
	Acurácia
	Interoperabilidade
	Segurança
	Conformidade relacionada à funcionalidade
Confiabilidade	Maturidade
	Tolerância a falhas
	Recuperabilidade
	Conformidade relacionada à confiabilidade
Usabilidade	Compreensibilidade
	Apreensibilidade
	Operacionalidade
	Atratividade
	Conformidade relacionada à usabilidade
Eficiência	Comportamento em relação ao tempo
	Comportamento em relação aos recursos
	Conformidade relacionada à eficiência
Manutenibilidade	Analisabilidade
	Modificabilidade
	Estabilidade
	Testabilidade
	Conformidade relacionada à manutenibilidade
Portabilidade	Adaptabilidade
	Capacidade para ser instalado
	Coexistência
	Capacidade para substituir
	Conformidade relacionada à portabilidade

2.3.2 ISO/IEC 25000 – Projeto SQuaRE

Em 1999, o comitê técnico da ISO iniciou a revisão das normas ISO/IEC 9126 e a ISO/IEC 14598, realizando a convergência dessas em uma única família. Isso foi realizado para eliminar falhas, ambigüidades e conflitos presentes. A família dessa norma é chamada ISO/IEC 25000 (ISO/IEC, 2005) também chamada de projeto SQuaRE (*Software Product Quality Requirements and Evaluation*). O objetivo geral é a confecção de uma família de normas logicamente organizada, visando cobrir dois processos principais: a especificação de requisitos e a avaliação da qualidade de software, ambos utilizando-se de um processo de medição.

Por se tratar de uma família de normas, o projeto SQuaRE foi definido dentro de um intervalo numérico. Esse está compreendido entre os números 25000 e 25099. Um processo de avaliação de qualidade deve contemplar as várias etapas existentes dentro do ciclo de vida de um software, a norma realizou a divisão deste intervalo. Em maio de 2002, a numeração final das divisões foi aprovada e aplicada. As divisões da família SQuaRE são as seguintes, conforme ilustra a Figura 2-1:

- **ISO/IEC 2500n**: Divisão da Gestão da Qualidade – Os padrões descritos como modelos, termos e definições, são utilizados em todos os outros padrões. Enumera os caminhos a serem usados pelo gerente do produto.
- **ISO/IEC 2501n**: Divisão do Modelo de Qualidade – Apresentam modelos de qualidade para software, sistemas em uso e dados.
- **ISO/IEC 2502n**: Divisão de Métricas de Qualidade – Apresentam modelos e definições matemáticas de métricas de qualidade.
- **ISO/IEC 2503n**: Divisão de Requisitos de Qualidade – Ajudam a definir requisitos de qualidade para utilização nos sistemas.
- **ISO/IEC 2504n**: Divisão da Avaliação da Qualidade – Apresentam requisitos, recomendações e guias para a avaliação de um software.
- **ISO/IEC 25050 – ISO/IEC 25099**: Extensão de padrões SQuaRE – Incluem informações como requisitos de qualidade para software COTS e padrões de relatórios utilizados pela indústria.

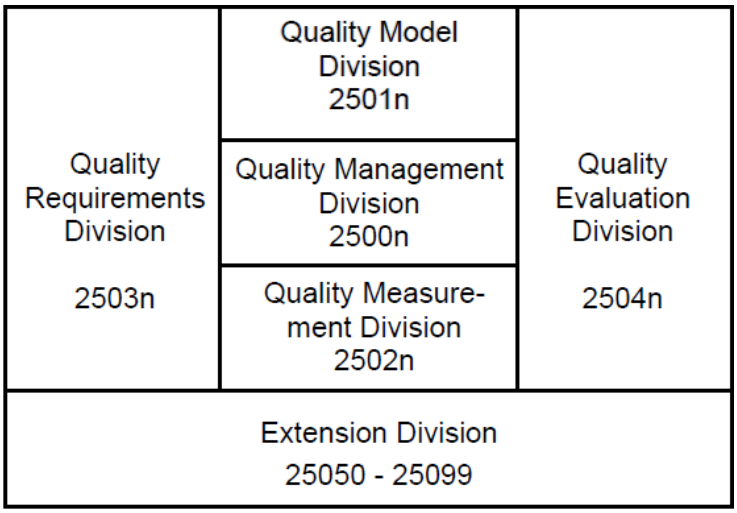

Figura 2-1. Arquitetura atual da série ISO/IEC 25000, adaptado de (ISO/IEC, 2005).

A norma ISO/IEC 25010 (ISO/IEC, 2011) apresenta o modelo de qualidade para software, com o nome *Product Quality Model*, atualizando o modelo definido na norma ISO/IEC 9126. Esse modelo auxilia a avaliação por vários atores de um projeto de software como desenvolvedores, gerentes, compradores e integradores de componentes e o usuário. A maioria das mudanças está relacionada com as características e subcaracterísticas do modelo de qualidade. Essas mudanças são:

- Característica **Funcionalidade**:
  - Mudança de nome **Funcionalidade** para **Adequação Funcional**.

    - Adição da subcaracterística de **Completude Funcional**.
    - Mudança de nome **Acurácia** para **Corretude Funcional**.
    - Mudança de nome **Adequação** para **Adequação Funcional**.
    - Mudança da subcaracterística **Interoperabilidade** para a característica **Compatibilidade**.
    - A subcaracterística **Segurança** passa a ser uma característica.
- Característica **Eficiência**:
    - Mudança de nome **Eficiência** para **Eficiência de Performance**.
    - Adição da subcaracterística **Capacidade**.
- Adição da Característica **Compatibilidade**:
    - Adição da subcaracterística **Coexistência**.
    - Adição da subcaracterística **Interoperabilidade**.
- Característica **Usabilidade**:
    - Adição da subcaracterística **Proteção a Erros do Usuário**.
    - Adição da subcaracterística **Acessibilidade**.
    - Mudança de Nome **Atratividade** para **Estética de Interface com o Usuário**.
- Característica **Confiabilidade**:
    - Adição da subcaracterística **Disponibilidade**.
- Adição da Característica **Segurança**:
    - Adição da subcaracterística **Confidencialidade**.
    - Adição da subcaracterística **Integridade**.
    - Adição da subcaracterística **Não-Repúdio**.
    - Adição da subcaracterística **Responsabilização**.
    - Adição da subcaracterística **Autenticidade**.
- Característica **Manutenibilidade**:
    - Adição da subcaracterística **Modularidade**.
    - Adição da subcaracterística **Reusabilidade**.
    - Mudança de Nome **Estabilidade** para **Modificabilidade**.
- Característica **Portabilidade**:
    - Mudança da subcaracterística **Coexistência** para a característica **Portabilidade**.

Assim o modelo tem oito características e 31 subcaracterísticas, conforme Quadro 2-2. Essa norma é o padrão para qualificação de qualquer tipo de software,

mas para esse trabalho é necessário abordar os Modelos que descenderam dessas normas, especificamente para o universo de componentes de software.

Quadro 2-2. Características e Subcaracterísticas da ISO/IEC 25010

CARACTERÍSTICA	SUBCARACTERÍSTICA
Adequação Funcional	Completude Funcional
	Corretude Funcional
	Adequação Funcional
Confiabilidade	Maturidade
	Tolerância a falhas
	Recuperabilidade
	Disponibilidade
Usabilidade	Conhecimento Adequado
	Apreensibilidade
	Operacionalidade
	Acessibilidade
	Proteção de Erro de Usuário
	Estética de Interface com o Usuário
Eficiência de Performance	Comportamento em relação ao tempo
	Comportamento em relação aos recursos
	Capacidade
Manutenibilidade	Analisabilidade
	Modificabilidade
	Modularidade
	Testabilidade
	Reusabilidade
Portabilidade	Adaptabilidade
	Instalabilidade
	Substituibilidade
Segurança	Confidencialidade
	Integridade
	Não-Repúdio
	Responsabilização
	Autenticidade
Compatibilidade	Coexistência
	Interoperabilidade

2.4 Modelos de Qualidade para Componentes de Software

Apesar de definir modelos de qualidade para softwares as normas ISO/IEC 9126 e ISO/IEC 25010 não são focadas em componentes de software, tornando-se pontos de partida para pesquisadores definirem seus modelos, realizando adaptações de acordo com o universo de componentes. Bertoa e Vallecillo (2002) descrevem outras dificuldades para determinar um padrão de qualidade:

> "Nós achamos que a avaliação de qualidade de componentes de software precisa ser realizada por institutos independentes, ao menos enquanto os vendedores de software não atingirem o nível de maturidade que os vendedores de hardware atualmente têm ." (p. 11, tradução nossa)

A literatura existente sobre modelos de qualidade para componentes não possui muitos relatos de experiências práticas com a aplicação dos modelos. Alvaro ET AL (2009) complementa que a literatura pode ser dividida em dois períodos, de 1993 a 2001 onde o foco era sobre os modelos baseados em teste e modelos matemáticos e pós 2001 onde o foco passou a ser sobre as técnicas e modelos baseados na previsão de requisitos de qualidade. O foco deste trabalho se encontra no segundo período.

2.4.1 Modelo de Qualidade de (BERTOA; VALLECILLO, 2002)

Modelo proposto por Manuel Bertoa e Antonio Vallecillo em 2002 no 6º International Workshop on Quantitative Approaches in Object-Oriented Software Engineering. Foi um dos primeiros modelos específico para componentes, adaptando a norma ISO/IEC 9126, tendo como foco os componentes COTS. O Quadro 2-3 apresenta o modelo de qualidade. Em resumo, as modificações realizadas foram as seguintes:

- Remoção da característica de ***Portabilidade***
- Remoção da subcaracterística ***Tolerância a Falhas***
- Remoção das subcaracterísticas ***Analisabilidade*** e ***Estabilidade***
- Adição da subcaracterística ***Compatibilidade*** na característica de ***Funcionalidade***. Verifica a compatibilidade com versões anteriores.
- Adição da subcaracterística ***Complexidade*** na característica de ***Usabilidade***. Verifica a dificuldade da integração em um sistema.
- Ocorreram mudanças no significado das subcaracterísticas de ***Apreensibilidade***, ***Compreensibilidade*** e ***Operacionalidade*** da característica de ***Usabilidade***, passando a atuar de forma distinta.

O modelo define atributos para cada subcaracterística e alguns tipos de métricas para a avaliação. Os tipos de métricas definidos são os seguintes:

- **Presença**: Utilizada para indicar se um determinado atributo está presente dentro do componente e sua implementação. Consiste de um valor booleano e uma string representando os itens citados.
- **Tempo**: Utilizada para indicar intervalos de tempo. Consiste em um inteiro que armazena o valor absoluto, ou numérico e uma String que indica a unidade de tempo (Segundos, minutos, horas, etc.).

- **Nível**: Utilizada para descrever níveis. Consiste em um inteiro que armazena um valor que vai do 0, muito baixo, até o 4, muito alto. Os outros valores são 1, baixo, 2, médio, e 3, alto.
- **Proporção**: Utilizada para descrever níveis de medição em porcentagens. Consiste em um valor inteiro que varia do valor 0 ao valor 100.

Quadro 2-3. Modelo de Qualidade para Componentes de (BERTOA; VALLECILLO, 2002)

CARACTERÍSTICA	SUBCARACTERÍSTICA
Funcionalidade	Adequação
	Precisão
	Interoperabilidade
	Conformidade
	Segurança
	Compatibilidade
Confiabilidade	Maturidade
	Recuperabilidade
Usabilidade	Apreensibilidade
	Compreensibilidade
	Operacionalidade
	Complexidade
Eficiência	Comportamento em relação ao tempo
	Comportamento em relação aos recursos
Manutenibilidade	Variabilidade
	Testabilidade

O modelo apresentado tem pontos positivos por apresentar atributos e métricas que realizam a avaliação de um componente. Mas foi uma mudança pequena de características para o universo de componentes, negligenciando outros itens como interfaces, reúso, segurança. Contudo foi o primeiro passo para a construção dos novos modelos.

2.4.2 Modelo de Qualidade de (SIMÃO; BELCHIOR, 2002)

Modelo proposto por Régis Patrick e Arnaldo Belchior em 2002 no 1º Simpósio Brasileiro de Qualidade de Software. O modelo realizou uma adaptação da norma ISO/IEC 9126, considerando alguns itens descritos pelo modelo de Bertoa e Vallecillo (2002), abordando qualquer tipo de componente. O Quadro 2-4 apresenta o modelo de qualidade. Em resumo, as modificações realizadas foram as seguintes:

- Não foi removida nenhuma característica e subcaracterística da Norma ISO/IEC 9126.

- Adição da subcaracterística ***Autocontido*** na característica de ***Funcionalidade***. Verifica se um componente é capaz de realizar, de forma completa, a função que ele desempenha.
- Adição da subcaracterística ***Coesão Funcional*** na característica de ***Funcionalidade***. Verifica se um componente possui todos os seus elementos voltados ao desempenho dos serviços oferecidos.
- Adição da subcaracterística ***Avaliabilidade*** na característica de ***Confiabilidade***. Verifica se um componente pode ser avaliado com relação a sua forma e conteúdo.
- Adição da subcaracterística ***Acessibilidade*** na característica de ***Usabilidade***. Verifica se um componente pode ser facilmente localizado, identificado e acessado.
- Adição da subcaracterística ***Legibilidade*** na característica de ***Usabilidade***. Verifica se um componente é de fácil compreensão.
- Adição da subcaracterística ***Facilidade de Uso*** na característica de ***Usabilidade***. Verifica a facilidade de utilizar o componente.
- Adição da subcaracterística ***Comportamento em Relação ao Estado*** na característica de ***Eficiência***. Verifica se o componente consegue preservar, transportar e recuperar seu estado.
- Adição da subcaracterística ***Escalabilidade*** na característica de ***Eficiência***. Verifica se o componente acomoda maiores volumes de processos sem a necessidade de mudanças em sua implementação.
- Adição da subcaracterística ***Nível de Granularidade Adequado*** na característica de ***Eficiência***. Verifica se o componente foi desenvolvido com o nível de granularidade adequado para ser útil, atualizável, manutenível e possuir alta performance
- Adição da subcaracterística ***Implementabilidade*** na característica de ***Manutenibilidade***. Verifica se o componente pode ser implementado em função da disponibilidade de recursos.
- Mudança do nome ***Capacidade para Substituir*** para ***Substituibilidade***

O modelo define três dimensões nas quais o grau de importância para as características e subcaracterísticas pode variar: *Domínio de Aplicação*, domínio onde o componente é utilizado; *Função Arquitetural*, qual camada o componente está

dentro do sistema, interface, negócio, dados, infraestrutura; e, *Nível de Abstração*, a utilização do componente nas formas de especificação, projeto e código.

Não foram definidas métricas para a avaliação de cada subcaracterística, mas foram definidas características internas destas. A forma de avaliação do modelo ocorreu através de survey com especialistas e o cálculo de quais características eram mais importantes foi realizado via *fuzzy logic*.

Quadro 2-4. Modelo de Qualidade para Componentes de (SIMÃO; BELCHIOR, 2002)

CARACTERÍSTICA	SUBCARACTERÍSTICA
Funcionalidade	Adequação
	Precisão
	Autocontido
	Coesão Funcional
	Interoperabilidade
	Segurança
	Conformidade com a Funcionalidade
Confiabilidade	Maturidade
	Tolerância a falhas
	Recuperabilidade
	Avaliabilidade
	Conformidade com a Confiabilidade
Usabilidade	Acessibilidade
	Legibilidade
	Compreensibilidade
	Facilidade de Uso
	Apreensibilidade
	Operacionalidade
	Atratividade
	Conformidade com a Usabilidade
Eficiência	Comportamento em Relação ao Tempo
	Comportamento em Relação aos Recursos
	Comportamento em Relação ao Estado
	Escalabilidade
	Nível de Granularidade Adequado
	Conformidade com a Eficiência
Manutenibilidade	Analisabilidade
	Implementabilidade
	Modificabilidade
	Estabilidade
	Testabilidade
	Conformidade com a Manutenibilidade
Portabilidade	Adaptabilidade
	Capacidade para ser Instalado
	Coexistência
	Substituibilidade
	Conformidade com a Portabilidade

2.4.3 Modelo de Qualidade de (RAWASHDEH; MATALKAH, 2006)

Modelo proposto por Adnan Rawashdeh e Bassem Matalkah em 2006, publicado no *Journal of Computer Science*. Realiza uma adaptação de vários modelos como a Norma ISO/IEC 9126, a Norma ISO/IEC 15504-2, o modelo de qualidade de Dromey (1995) e o modelo de qualidade de Bertoa e Vallecillo (2002). O modelo tem foco nos componentes COTS. O Quadro 2-5 apresenta o modelo de qualidade. Em resumo, as modificações realizadas foram as seguintes:

- Remoção da característica de ***Portabilidade***
- Remoção das subcaracterísticas ***Tolerância a Falhas*** e ***Conformidade Relacionada à Confiabilidade***
- Remoção das subcaracterísticas ***Atratividade*** e ***Conformidade Relacionada à Usabilidade***
- Remoção da subcaracterística ***Conformidade Relacionada à Eficiência***
- Remoção das subcaracterísticas ***Analisabilidade, Modificabilidade, Estabilidade*** e ***Conformidade Relacionada à Manutenibilidade***
- Adição da subcaracterística ***Compatibilidade*** na característica de ***Funcionalidade***. Verifica a compatibilidade com versões anteriores.
- Adição da subcaracterística ***Complexidade*** na característica de ***Usabilidade***. Verifica a dificuldade de integração em um sistema.
- Adição da subcaracterística ***Variabilidade*** na característica de ***Manutenibilidade***. Verifica se o componente pode ser variado, identificando seus mecanismos de variabilidade.
- Adição da característica ***Gerenciamento***.
- Adição da subcaracterística **Gestão de Qualidade** na característica de **Gerenciamento**. Indica quais pessoas dentro da organização estão fazendo a parte de gestão de qualidade de um componente.

Um item interessante do modelo está na parte de stakeholders, isto é, qualquer pessoa ou grupo que será afetado pelo sistema, direta ou indiretamente. Existem cinco categorias: usuário final do sistema, analistas de software, analistas de qualidade, gerente de projeto, proprietário do software. Estas formam dois grupos de classificação de características.

Quadro 2-5. Modelo de Qualidade para Componentes de (RAWASHDEH; MATALKAH, 2006)

CARACTERÍSTICA	SUBCARACTERÍSTICA
Funcionalidade	Precisão
	Segurança
	Adequação
	Interoperabilidade
	Conformidade
	Compatibilidade
Confiabilidade	Recuperabilidade
	Maturidade
Usabilidade	Apreensibilidade
	Compreensibilidade
	Operacionalidade
	Complexidade
Eficiência	Comportamento em relação ao tempo
	Comportamento em relação aos recursos
Manutenibilidade	Variabilidade
	Testabilidade
Gerenciamento	Gestão da Qualidade

O primeiro grupo é formado por usuário final do sistema, analistas de software e analistas de qualidade, responsáveis pelas características de Funcionalidade, Confiabilidade, Usabilidade e Eficiência. A sua atividade está no desenvolvimento do software. O segundo é formado por gerente de projeto e proprietário do software responsáveis pelas características de Manutenibilidade e Gerenciamento. A sua atividade está no gerenciamento e controle de prazos

Esse modelo representa apenas uma adaptação, pois características importantes como reusabilidade e tolerância à faltas foram removidas do modelo. Outro ponto negativo é a falta da apresentação de métricas para a avaliação das subcaracterísticas do modelo, dificultando a realização da avaliação.

2.4.4 Modelo de Qualidade de (ANDREOU; TZIAKOURIS, 2007)

Modelo proposto por Andreas Andreou e Marios Tziakouris em 2007, publicado no *Journal Information and Software Technology*. Realiza uma adaptação da Norma ISO/IEC 9126 e tem foco nos componentes originais. O Quadro 2-6 apresenta o modelo de qualidade. Em resumo, as modificações realizadas foram as seguintes:

- Remoção da característica de ***Portabilidade***
- Remoção das subcaracterísticas ***Adequação***, ***Acurácia*** e ***Conformidade Relacionada à Funcionalidade***

- Remoção das subcaracterísticas ***Maturidade***, ***Tolerância a Falhas***, ***Recuperabilidade*** e ***Conformidade Relacionada à Confiabilidade***
- Remoção das subcaracterísticas ***Compreensibilidade***, ***Atratividade*** e ***Conformidade Relacionada à Usabilidade***
- Remoção das subcaracterísticas ***Comportamento em Relação ao Tempo***, ***Comportamento em Relação aos Recursos*** e ***Conformidade Relacionada à Eficiência***
- Remoção das subcaracterísticas ***Analisabilidade, Modificabilidade, Estabilidade*** e ***Conformidade Relacionada à Manutenibilidade***
- Adição da subcaracterística ***Plenitude*** na característica de ***Funcionalidade***. Verifica o quão bem o componente se encaixa nos requisitos de usuário e verifica se o serviço oferecido é satisfatório ou não.
- Adição da subcaracterística ***Serviço de Estabilidade*** na característica de ***Confiabilidade***. Verifica se um componente está livre de erros e a habilidade de se recuperar após uma falha,
- Adição da subcaracterística ***Conjunto de Dados*** na característica de ***Confiabilidade***. Verifica se os dados retornados pelo componente estão corretos e com qualidade.
- Adição da subcaracterística ***Ferramentas de Ajuda*** na característica de ***Usabilidade***. Verifica se um componente possui um manual para melhor forma de uso assim como a efetividade da ferramenta de ajuda.
- Adição da subcaracterística ***Identificabilidade – Acessibilidade*** na característica de ***Usabilidade***. Verifica se um componente pode ser identificado e obtido para a utilização.
- Adição da subcaracterística ***Tempo de Resposta*** na característica de ***Eficiência***. Verifica a diferença de tempo entre a requisição e a resposta do comando do componente.
- Adição da subcaracterística ***Atraso do Sistema*** na característica de ***Eficiência***. Avalia os recursos do sistema necessários para realizar a execução do componente.
- Adição da subcaracterística ***Variabilidade*** na característica de ***Manutenibilidade***. Verifica a facilidade e o esforço necessário para modificar o componente.

- Adição da subcaracterística ***Personalização*** na característica de ***Manutenibilidade***. Avalia a habilidade do componente de ser customizado de acordo com as necessidades do usuário.

O modelo define atributos para cada subcaracterística e alguns tipos de métricas para a avaliação. Os tipos de métricas definidos são os seguintes:

- **Proporção**: Utilizada para descrever níveis de medição em porcentagens. Consiste em um valor inteiro que varia do valor 0 ao valor 100.
- **Listagem:** Utilizada para descrever os tipos de algum item, como por exemplo, lista de tipos de dispositivos suportados.
- **Presença**: Utilizada para indicar se um determinado atributo está presente dentro do componente e como está implementado. Consiste de um valor booleano e uma string representando os itens citados.
- **Tempo**: Utilizada para indicar intervalos de tempo. Esse intervalo consiste em um inteiro que armazena o valor absoluto, ou numérico e uma String que indica a unidade de tempo (Segundos, minutos, horas, etc.).
- **Nível**: Utilizada para descrever níveis de medição. Esse nível consiste em um inteiro que armazena um valor que vai do 1, muito baixo, até o 5, muito alto. Os outros valores são 2, baixo, 3, médio, e 4, alto.

Quadro 2-6. Modelo de Qualidade para Componentes de (ANDREOU; TZIAKOURIS, 2007)

CARACTERÍSTICA	SUBCARACTERÍSTICA
Funcionalidade	Interoperabilidade
	Plenitude
	Segurança
Confiabilidade	Serviço de Estabilidade
	Conjunto de Resultados
Usabilidade	Apreensibilidade
	Ferramentas de Ajuda
	Operacionalidade
	Identificabilidade – Acessibilidade
Eficiência	Tempo de Resposta
	Atraso do Sistema
Manutenibilidade	Variabilidade
	Testabilidade
	Personalização

O modelo apresentado tem pontos positivos como a definição de várias métricas, uma boa descrição e experimentos. Mas não trata algumas características

de componentes como a portabilidade e a reusabilidade e não realiza o aprofundamento na subcaracterística de segurança, deixada em segundo plano.

2.4.5 Modelo de Qualidade de (COLOMBO ET AL., 2007)

Modelo proposto por Regina Colombo, Ana Guerra, Maria Aguayo e Darley Peres em 2007 na Conferência IADIS Ibero-Americana WWW/Internet 2007. O modelo realizou uma adaptação da Norma ISO/IEC 9126, e do modelo de Bertoa e Vallecillo (2002), sendo utilizado para qualquer tipo de componente. O Quadro 2-7 apresenta o modelo de qualidade. Em resumo, as modificações realizadas foram as seguintes:

- Remoção da subcaracterística ***Conformidade Relacionada à Funcionalidade***
- Remoção das subcaracterísticas ***Tolerância a Falhas*** e ***Conformidade Relacionada à Confiabilidade***
- Remoção das subcaracterísticas ***Compreensibilidade***, ***Apreensibilidade***, ***Atratividade*** e ***Conformidade Relacionada à Usabilidade***
- Remoção da subcaracterística ***Conformidade Relacionada à Eficiência***
- Remoção das subcaracterísticas ***Analisabilidade***, ***Estabilidade*** e ***Conformidade Relacionada à Manutenibilidade***
- Remoção das subcaracterísticas ***Adaptabilidade***, ***Capacidade para ser Instalado***, ***Coexistência*** e ***Conformidade Relacionada à Portabilidade***
- Adição da característica ***Completude***
- Adição da subcaracterística ***Identificador*** na característica de ***Completude***. Verifica se a descrição do componente contém identificações importantes
- Adição da subcaracterística ***Declarações*** na característica de ***Completude***. Verifica se a descrição do componente contém declarações sobre suporte e manutenção
- Adição da subcaracterística ***Declaração de Funcionalidade*** na característica de ***Completude***. Verifica se na descrição do componente, existem declarações sobre as funções, valores-limites, grau de precisão, interoperabilidade e segurança de acesso do componente

- Adição da subcaracterística ***Declaração de Confiabilidade*** na característica de ***Completude***. Verifica se na descrição do componente, existem informações sobre procedimentos para a preservação dos dados
- Adição da subcaracterística ***Declaração de Usabilidade*** na característica de ***Completude***. Verifica se na descrição do componente, existem declarações, adaptações e proteções sobre o conhecimento específico requerido do usuário
- Adição da subcaracterística ***Declaração de Eficiência*** na característica de ***Completude***. Verifica se na descrição do componente, existem dados sobre o comportamento do componente em relação ao tempo
- Adição da subcaracterística ***Declaração de Manutenibilidade*** na característica de ***Completude***. Verifica se na descrição do componente, existem declarações sobre manutenibilidade
- Adição da subcaracterística ***Declaração de Portabilidade*** na característica de ***Completude***. Verifica se na descrição do componente, existem declarações sobre portabilidade
- Adição da subcaracterística ***Declaração de Qualidade de Uso*** na característica de ***Completude***. Verifica se na descrição do componente, existem declarações de qualidade sob a perspectiva do desenvolvedor, projetista, arquiteto de software
- Adição da característica ***Conteúdo***
- Adição da subcaracterística ***Avaliação de Adequação*** na característica de ***Conteúdo***. Verifica se o conteúdo da descrição do componente é inteligível, completa e possui uma boa organização
- Adição da subcaracterística ***Consistência*** na característica de ***Conteúdo***. Verifica se o conteúdo da descrição do componente é livre de inconsistências internas
- Adição da subcaracterística ***Termos Técnicos*** na característica de ***Conteúdo***. Verifica se os termos técnicos possuem significados únicos
- Adição da subcaracterística ***Declarações Corretas*** na característica de ***Conteúdo***. Verifica se a descrição do componente possui declarações corretas

- Adição da subcaracterística ***Declarações Testáveis*** na característica de ***Conteúdo***. Verifica se a descrição do componente possui declarações passíveis de serem testadas
- Adição da subcaracterística ***Completitude*** na característica de ***Funcionalidade***. Verifica o quão completa é a documentação das funcionalidades do componente
- Adição da subcaracterística ***Conformidade*** na característica de ***Funcionalidade***. Verifica a capacidade do componente de estar de acordo com padrões regulamentados
- Adição da subcaracterística ***Apresentação e Organização*** na característica de ***Usabilidade***. Verifica se a documentação do componente possui uma boa apresentação e organização
- Mudança do nome ***Acurácia*** para ***Precisão***

Quadro 2-7. Modelo de Qualidade para Componentes de (COLOMBO ET AL., 2007)

CARACTERÍSTICA	SUBCARACTERÍSTICA
Completitude	Identificador
	Declarações
	Declaração de Funcionalidade
	Declaração de Confiabilidade
	Declaração de Usabilidade
	Declaração de Eficiência
	Declaração de Manutenibilidade
	Declaração de Portabilidade
	Declaração de Qualidade de Uso
Conteúdo	Avaliação de Adequação
	Consistência
	Termos Técnicos
	Declarações Corretas
	Declarações Testáveis
Funcionalidade	Completitude
	Adequação
	Interoperabilidade
	Conformidade
	Precisão
	Segurança
Confiabilidade	Maturidade
	Recuperabilidade
Usabilidade	Apresentação e Organização
	Operacionalidade
Eficiência	Comportamento em relação ao tempo
	Comportamento em relação aos recursos
Manutenibilidade	Modificabilidade
	Testabilidade
Portabilidade	Capacidade para Substituir

O modelo define alguns atributos internos dentro das subcaracterísticas para a avaliação, mas não existe métricas para avaliar os atributos internos, como em outros modelos. O modelo trabalha a Norma ISO/IEC 9126, embasando as suas características e subcaracterísticas fortemente na documentação do componente. Caso a documentação não esteja detalhada, torna-se difícil realizar a avaliação.

2.4.6 Modelo de Qualidade de (CHOI ET AL., 2008)

Modelo proposto por Yoonjung Choi, Sungwook Lee, Houp Song, Jingoo Park, SunHee Kim em 2008 no The 10º *International Conference on Advanced Communication Technology*. Realiza uma adaptação da norma ISO/IEC 9126 e pode ser aplicado para qualquer tipo de componente. Além disso, é definido um nível especializado de avaliação para cada tipo de componente. O Quadro 2-8 apresenta o modelo de qualidade. Em resumo, as modificações realizadas foram as seguintes:

- Remoção das subcaracterísticas ***Acurácia***, ***Interoperabilidade*** e ***Conformidade Relacionada à Funcionalidade***
- Remoção da subcaracterística ***Conformidade Relacionada à Confiabilidade***
- Remoção das subcaracterísticas ***Atratividade*** e ***Conformidade Relacionada à Usabilidade***
- Remoção da subcaracterística ***Conformidade Relacionada à Eficiência***
- Remoção das subcaracterísticas ***Analisabilidade, Modificabilidade, Estabilidade*** e ***Conformidade Relacionada à Manutenibilidade***
- Remoção das subcaracterísticas ***Capacidade para ser Instalado***, ***Coexistência*** e ***Conformidade Relacionada à Portabilidade***
- Adição da subcaracterística ***Plenitude*** na característica de ***Funcionalidade***. Verifica o grau de coerência entre os requisitos e os produtos de trabalho relacionados.
- Adição da subcaracterística ***Escalabilidade*** na característica de ***Eficiência***. Verifica se o componente acomoda maiores volumes de processos sem a necessidade de mudanças em sua implementação.

- Adição da subcaracterística ***Variabilidade*** na característica de ***Manutenibilidade***. Verifica se o componente pode ser variado, identificando seus mecanismos de variabilidade.
- Adição da característica ***Reusabilidade***.
- Adição da subcaracterística ***Comunalidade Funcional*** na característica de ***Reusabilidade***. Verifica o nível de funcionalidades comuns dentro do componente.
- Adição da subcaracterística ***Personalização*** na característica de ***Reusabilidade***. Avalia a habilidade do componente de ser customizado de acordo com as necessidades do usuário.
- Adição da subcaracterística ***Conformidade com o Modelo de Domínio*** na característica de ***Reusabilidade***. Verifica se o componente está de acordo com o domínio do software.
- Adição da subcaracterística ***Conformidade com o Modelo de Componentes*** na característica de ***Reusabilidade***. Verifica se o componente está de acordo com o modelo de componentes.
- Adição da característica ***Modularidade***.
- Adição da subcaracterística ***Coesão*** na característica de ***Modularidade***. Verifica se as responsabilidades do componente estão corretas.
- Adição da subcaracterística ***Acoplamento*** na característica de ***Modularidade***. Verifica o nível de dependência entre componentes.
- Mudança do nome ***Capacidade para Substituir*** para ***Substituibilidade***

O modelo define atributos para cada subcaracterística e métricas de proporção para cada atributo. Para cada subcaracterística é realizada uma média das métricas de proporção. É descrito no artigo que foi realizado um teste piloto com três componentes de multimídia para a TV Digital com duração de dez dias úteis com resultados satisfatórios. Essa média é analisada em uma escala de classificação, que tem os seguintes valores:

- Entre 0% e 15% – N (*Not Achieved*) – Não alcançada
- Entre 15% e 50% – P (*Partially Achieved*) – Parcialmente alcançada
- Entre 50% e 85% – L (*Largely Achieved*) – Largamente alcançada
- Entre 85% e 100% – F(*Fully Achieved*) – Totalmente alcançada

Quadro 2-8. Modelo de Qualidade para Componentes de (CHOI ET AL., 2008)

CARACTERÍSTICA	SUBCARACTERÍSTICA
Funcionalidade	Adequação
	Plenitude
	Segurança
Confiabilidade	Maturidade
	Tolerância a falhas
	Recuperabilidade
Usabilidade	Compreensibilidade
	Apreensibilidade
	Operacionalidade
Eficiência	Comportamento em relação ao tempo
	Comportamento em relação aos recursos
	Escalabilidade
Manutenibilidade	Testabilidade
	Variabilidade
Portabilidade	Substituibilidade
	Adaptabilidade
Reusabilidade	Comunalidade Funcional
	Personalização
	Conformidade com o Modelo de Domínio
	Conformidade com o Modelo de Componentes
Modularidade	Coesão
	Acoplamento

2.4.7 Modelo de Qualidade de (SHARMA; KUMAR; GROVER, 2008)

Modelo proposto por Arun Sharma, Rajesh Kumar e P. S. Grover em 2008 no *Journal ACM SIGSOFT Software Engineering Notes*. O modelo realizou uma adaptação da Norma ISO/IEC 9126, do modelo de Boehm, Brown e Lipow (1976), do modelo de McCall, Richards e Walters (1977) e do modelo de Dromey (1995) abordando qualquer tipo de componente. O Quadro 2-9 apresenta o modelo de qualidade. Em resumo, as modificações realizadas foram as seguintes:

- Remoção da subcaracterística ***Conformidade Relacionada à Funcionalidade***
- Remoção da subcaracterística ***Conformidade Relacionada à Confiabilidade***
- Remoção da subcaracterística ***Conformidade Relacionada à Usabilidade***
- Remoção da subcaracterística ***Conformidade Relacionada à Eficiência***
- Remoção da subcaracterística ***Conformidade Relacionada à Manutenibilidade***
- Remoção da subcaracterística ***Conformidade Relacionada à Portabilidade***

- Adição da subcaracterística ***Reusabilidade*** na característica de ***Funcionalidade***. Verifica o grau de reúso do componente.
- Adição da subcaracterística ***Complexidade*** na característica de ***Usabilidade***. Verifica a dificuldade de integração em um sistema.
- Adição da subcaracterística ***Escalabilidade*** na característica de ***Eficiência***. Verifica se o componente acomoda maiores volumes de processos sem a necessidade de mudanças em sua implementação.
- Adição da subcaracterística ***Variabilidade*** na característica de ***Manutenibilidade***. Verifica se o componente pode ser variado, identificando seus mecanismos de variabilidade.
- Adição da subcaracterística ***Rastreabilidade*** na característica de ***Manutenibilidade***. Verifica se o componente é compatível com as versões antigas do componente.
- Mudança do nome ***Modificabilidade*** para ***Flexibilidade***
- Mudança do nome ***Capacidade para ser instalado*** para ***Instalabilidade***
- Mudança do nome ***Capacidade para substituir*** para ***Substituibilidade***

Quadro 2-9. Modelo de Qualidade para Componentes de (SHARMA; KUMAR; GROVER, 2008)

CARACTERÍSTICA	SUBCARACTERÍSTICA
Funcionalidade	Adequação
	Acurácia
	Interoperabilidade
	Segurança
	Reusabilidade
Confiabilidade	Maturidade
	Tolerância a falhas
	Recuperabilidade
Usabilidade	Compreensibilidade
	Apreensibilidade
	Operacionalidade
	Atratividade
	Complexidade
Eficiência	Comportamento em relação ao tempo
	Comportamento em relação aos recursos
	Escalabilidade
Manutenibilidade	Estabilidade
	Analisabilidade
	Variabilidade
	Testabilidade
	Rastreabilidade
	Flexibilidade
Portabilidade	Adaptabilidade
	Instalabilidade
	Substituibilidade
	Coexistência

O modelo define métricas para realizar a avaliação de cada subcaracterística, mas diferente dos outros modelos, estas são unidades de medida únicas, ou seja, a unidade de medida base do modelo são valores normalizados em um intervalo de 0 a 1 universalizando as métricas e facilitando os cálculos de valores para os componentes, para gerar rankings.

2.4.8 Modelo de Qualidade de (ALVARO, 2009)

Modelo proposto por Alexandre Álvaro em sua dissertação de mestrado no ano de 2005, baseado na Norma ISO/IEC 9126. Em sua tese de Doutorado ele atualizou o seu modelo para a Norma ISO/IEC 25010. Utilizou também o modelo de Bertoa e Vallecillo (2002) e o modelo de Simão e Belchior (2002), abordando qualquer tipo de componente. O Quadro 2-10 apresenta o modelo de qualidade. Em resumo, as modificações realizadas foram as seguintes:

- Remoção da subcaracterística **Atratividade**
- Remoção das subcaracterísticas ***Analisabilidade*** e ***Modificabilidade***
- Remoção da subcaracterística ***Coexistência***
- Adição da subcaracterística ***Auto-Suficiência*** na característica de ***Funcionalidade***. Verifica se um componente é capaz de realizar, de forma completa, a função que ele desempenha.
- Adição da subcaracterística ***Configurabilidade*** na característica de ***Usabilidade***. Verifica o nível de configuração de um componente
- Adição da subcaracterística ***Escalabilidade*** na característica de ***Eficiência***. Verifica se o componente acomoda maiores volumes de processos sem a necessidade de mudanças em sua implementação.
- Adição da subcaracterística ***Variabilidade*** na característica de ***Manutenibilidade***. Verifica se o componente pode ser variado, identificando seus mecanismos de variabilidade.
- Adição da subcaracterística ***Reusabilidade*** na característica de ***Portabilidade***. Verifica o grau de reúso do componente
- Adição da característica ***Negociabilidade***.
- Adição da subcaracterística ***Preço*** na característica de ***Negociabilidade***. Indica qual é o preço do componente.

- Adição da subcaracterística ***Tempo para o mercado*** na característica de ***Negociabilidade***. Indica o tempo consumido para construir o componente e disponibilizá-lo no mercado.
- Adição da subcaracterística ***Mercado-alvo*** na característica de ***Negociabilidade***. Indica qual é o publico alvo do componente
- Adição da subcaracterística ***Licenciamento*** na característica de ***Negociabilidade***. Indica qual é a licença que o componente possui.
- Mudança do nome ***Capacidade para ser instalado*** para ***Deployabilidade***
- Mudança do nome ***Capacidade para substituir*** para ***Substituibilidade***

O modelo define atributos para cada subcaracterística e algumas métricas para a avaliação das subcaracterísticas. As métricas definidas são os seguintes:

- **Presença**: Utilizada para indicar se um determinado atributo está presente dentro do componente e como está implementado. Consiste de um valor booleano e uma string representando os itens citados.
- **IValues**: Utilizada para indicar o valor exato de uma característica em uma determinada unidade de medida. Esse intervalo consiste em um inteiro que armazena o valor absoluto ou numérico e uma String que indica a unidade de medida (Segundos, linhas de código, etc.).
- **Nível**: Utilizada para descrever níveis de medição. Esse nível consiste em um inteiro que armazena um valor que vai do 0, muito baixo, até o 4, muito alto. Os outros valores são 1, baixo, 2, médio, e 3, alto.
- **Proporção**: Utilizada para descrever níveis de medição em porcentagens. Consiste em um valor inteiro que varia do valor 0 ao valor 100.

O modelo dá um grande passo a frente, já que associa o modelo de Bertoa e Vallecillo (2002) adicionando subcaracterísticas compatíveis com o universo dos componentes. Apesar da comparação com a norma ISO/IEC 25010, algumas características definidas, como Segurança e Compatibilidade, não foram adequadamente estudadas e aplicadas. Outro ponto é a falta de uma série de testes mais elaborados.

Quadro 2-10. Modelo de Qualidade para Componentes de (ALVARO, 2009)

CARACTERÍSTICA	SUBCARACTERÍSTICA
Funcionalidade	Adequação
	Acurácia
	Interoperabilidade
	Segurança
	Auto-Suficiência
	Conformidade
Confiabilidade	Maturidade
	Recuperabilidade
	Tolerância a falhas
	Conformidade
Usabilidade	Compreensibilidade
	Configurabilidade
	Apreensibilidade
	Operacionalidade
	Conformidade
Eficiência	Comportamento em relação ao tempo
	Comportamento em relação aos recursos
	Escalabilidade
	Conformidade
Manutenibilidade	Estabilidade
	Variabilidade
	Testabilidade
	Conformidade
Portabilidade	Deployabilidade
	Substituibilidade
	Adaptabilidade
	Reusabilidade
	Conformidade
Negociabilidade	Preço
	Tempo para o mercado
	Mercado-alvo
	Licenciamento

2.4.9 Modelo de Qualidade de (KALAIMAGAL; SRINIVASAN, 2010)

Modelo proposto por Sivamuni Kalaimagal e Rengaramanujam Srinivasan em 2010 e publicado no *Journal ACM Sigsoft Software Engineering Notes.* O modelo realizou uma adaptação da norma ISO/IEC 25010, considerando também o modelo de Kalaimagal e Srinivasan (2010a), o modelo de Bertoa e Vallecillo (2002), o modelo de Rawashdeh e Matalkah (2006) e o modelo de Álvaro (2009), abordando componentes COTS e caixas pretas. O Quadro 2-11 apresenta o modelo de qualidade. Em resumo, as modificações realizadas foram as seguintes:

- Remoção das subcaracterísticas ***Adequação***, ***Interoperabilidade***, ***Segurança*** e ***Conformidade Relacionada à Funcionalidade***

- Remoção das subcaracterísticas ***Maturidade***, ***Tolerância a falhas*** e ***Conformidade Relacionada à Confiabilidade***
- Remoção das subcaracterísticas ***Compreensibilidade***, ***Atratividade*** e ***Conformidade Relacionada à Usabilidade***
- Remoção da subcaracterística ***Conformidade Relacionada à Eficiência***
- Remoção das subcaracterísticas ***Analisabilidade***, ***Modificabilidade***, ***Testabilidade*** e ***Conformidade Relacionada à Manutenibilidade***
- Remoção das subcaracterísticas ***Coexistência*** e ***Conformidade Relacionada à Portabilidade***
- Adição da subcaracterística ***Conformidade*** na característica de ***Funcionalidade***. Relação da aderência do componente aos padrões e convenções do projeto.
- Adição da subcaracterística ***Independência*** na característica de ***Funcionalidade***. Relação da funcionalidade do componente com o mínimo de suporte externo.
- Adição da subcaracterística ***Ajudabilidade*** na característica de ***Usabilidade***. Verifica se um componente possui um manual melhor uso assim como a efetividade da ferramenta de ajuda.
- Adição da subcaracterística ***Facilidade de Migração*** na característica de ***Manutenibilidade***. Relação do esforço necessário para migrar um componente de uma versão antiga do software para uma nova.
- Adição da subcaracterística ***Portabilidade*** na característica de ***Portabilidade***.
- Adição da característica ***Testabilidade***.
- Adição da subcaracterística ***Controle de Componente*** na característica de ***Testabilidade***. Verifica a facilidade de controle pelas operações.
- Adição da subcaracterística ***Documentação de Teste*** na característica de ***Testabilidade***. Relação com a existência de documentos de teste
- Adição da subcaracterística ***Rastreabilidade*** na característica de ***Testabilidade***. Relação com a capacidade de rastrear o status dos atributos do componente e seu comportamento.
- Adição da característica ***Segurança***.
- Adição da subcaracterística ***Autenticação*** na característica de ***Segurança***. Relação do acesso do componente as interfaces providas.

- Adição da subcaracterística **Criptografia de Dados** na característica de ***Segurança***. Relação do processo de proteção dos dados internos que estão contidos no componente.
- Adição da característica ***Interoperabilidade***.
- Adição da subcaracterística ***Compatibilidade*** na característica de ***Interoperabilidade***. Verifica a relação entre componentes de diferentes versões.
- Adição da característica ***Reusabilidade***.
- Adição da subcaracterística ***Generalidade*** na característica de ***Reusabilidade***. Relação com o conjunto de atributos que tornam o componente mais genérico.
- Adição da subcaracterística ***Independência de Hardware e Software*** na característica de ***Reusabilidade***. Verifica se o componente não possui nenhuma dependência de hardware ou software.
- Adição da subcaracterística ***Locabilidade*** na característica de ***Reusabilidade***. Relação com a facilidade de localizar o componente dentro de um repositório.
- Adição da característica ***Usabilidade em Uso***.
- Adição da subcaracterística ***Efetividade em Uso*** na característica de ***Usabilidade em Uso***. Relação com o quão efetivo um componente é.
- Adição da subcaracterística ***Satisfação em Uso*** na característica de ***Usabilidade em Uso***. Relação com os esforços de aprendizagem na aplicação.
- Adição da característica ***Segurança em Uso***.
- Adição da subcaracterística ***Risco do Software*** na característica de ***Segurança em Uso***. Relação com os riscos que podem ocorrer ao inserir o componente no software.
- Adição da subcaracterística ***Riscos Comerciais em Uso*** na característica de ***Segurança em Uso***. Relação com os riscos comerciais, como quebra de patentes, ao inserir o componente no software.
- Adição da subcaracterística ***Risco Operador em Uso*** na característica de ***Segurança em Uso***. Relação com os riscos que o operador ou usuário, pode causar ao sistema ao utilizar o componente inserido

- Adição da subcaracterística ***Risco em Uso Público*** na característica de ***Segurança em Uso***. Relação com os riscos que podem ocorrer ao utilizar o componente em lugares públicos.
- Mudança do nome ***Capacidade para ser instalado*** para ***Instalabilidade***
- Mudança do nome ***Capacidade para Substituir*** para ***Substituibilidade***

O modelo utiliza apenas um tipo de métrica, sendo esta um valor normalizado cujo intervalo encontra-se entre 0 e 1, igual a (SHARMA; KUMAR; GROVER, 2008). Existe a base sobre a norma ISO/IEC 25010, mas o modelo não aproveita todo o potencial da Norma, sendo somente um apêndice ao modelo pois subcaracterísticas como Tolerância a Falhas e Confidencialidade não foram inseridas.

Quadro 2-11. Modelo de Qualidade para Componentes de (KALAIMAGAL; SRINIVASAN, 2010)

CARACTERÍSTICA	SUBCARACTERÍSTICA
Funcionalidade	Conformidade
	Independência
	Acurácia
Confiabilidade	Recuperabilidade
Usabilidade	Ajudabilidade
	Apreensibilidade
	Operacionalidade
Eficiência	Comportamento em relação ao tempo
	Comportamento em relação aos recursos
Manutenibilidade	Estabilidade
	Facilidade de Migração
Portabilidade	Adaptabilidade
	Instalabilidade
	Portabilidade
	Substituibilidade
Testabilidade	Controle de Componente
	Documentação de Teste
	Rastreabilidade
Segurança	Autenticação
	Criptografia de Dados
Interoperabilidade	Compatibilidade
Reusabilidade	Generalidade
	Independência de Hardware e Software
	Locabilidade
Usabilidade em Uso	Efetividade em Uso
	Satisfação em Uso
Segurança em Uso	Risco do Software
	Riscos Comerciais em Uso
	Risco Operador em Uso
	Risco em Uso Público

Agora é necessário definir conceitos dentro da área de segurança de software, que será realizado pela próxima seção.

2.5 Aspectos de Segurança em Software

O primeiro ponto importante a ser definido dentro da segurança de software está na própria definição do termo segurança para software. Essa definição é realizada por (AVIZIENIS ET AL., 2004):

> "Segurança é uma composição de atributos de confidencialidade, integridade e disponibilidade, exigindo a existência concorrente de 1) disponibilidade somente para ações autorizadas, 2) confidencialidade, 3) integridade com o "impróprio" significando "Não Autorizado"." (p. 13, tradução nossa)

Para complementar essa definição é necessário definir conceitos relacionados com usuários de um sistema, agentes de software ou humanos. Avizienis ET AL (2004), define que os seguintes itens são relacionados com usuários:

- **Responsabilidade**, que ajuda a identificar a entidade que realizou determinada operação;
- **Autenticidade**, integridade da origem e do conteúdo de uma mensagem. Pode ser utilizada para garantir a integridade de outros atributos também;
- **Não-repúdio**, que não permite a refutação ou negação da responsabilidade sobre ações cometidas por uma entidade.

2.5.1 Normas de Segurança de Software

Existem diferentes formas de realizar a avaliação das características de segurança de um software. Uma proposta de Bayuk (2000) descreve que esta avaliação das características pode ser dividida em cinco categorias:

- Avaliação a partir de boas práticas do mercado (auditoria externa).
- Avaliação a partir de práticas de segurança internas (auditoria interna).
- Modelo de maturidade de capacitação.
- Análise de riscos.
- Eliminação de defeitos.

A auditoria externa consiste na busca por "melhores práticas" para realizar a proteção do software, comparando o nível de gerenciamento e o ambiente do sistema, desenvolvendo uma medida para a segurança. Essa é uma lista de pontos fracos do sistema que devem ser corrigidos para evitar riscos de segurança.

A primeira forma de auditoria externa surge com a norma TCSEC – *Trusted Computing System Evaluation Criteria* (DOD, 1985), ou *Orange Book*. Essa norma

possui o intuito de avaliar dispositivos de segurança, como itens de criptografia, firewalls, focando a garantia de confidencialidade das informações.

Em 1991, surge uma norma pelo grupo de países formados por França, Alemanha, Holanda e Reino Unido. Esta possui o nome de ITSEC - *Information Technology Security Evaluation Criteria* (ITSEC, 1991). A base é a avaliação de produtos e sistemas, separando as funcionalidades e a garantia de segurança.

Em 1992 o NIST - *National Institute of Standards and Technology* instituto estadunidense responsável por padrões de tecnologia. A norma chamada de MSFR - *Minimum Security Functionality Requirements for Multi-User Operating Systems* (MSFR, 1992) com o objetivo de enumerar um conjunto mínimo de requisitos funcionais de segurança para sistemas operacionais multiusuários, especialmente os relacionados com a garantia de qualidade para os processos de desenvolvimento dos fornecedores estadunidenses. Essa norma foi baseada no TCSEC.

Como cada país ou conglomerado com grande poder econômico escreveu uma norma própria, criou-se um problema para empresas multinacionais pois tinham de se adequar à legislação local para atender a clientes distintos. Assim a ISO/IEC apresentou a norma ISO/IEC 15408 em 1999, cujo nome é CC - *Common Criteria*. A versão mais atual dessa norma é a 3.1 de 2009 (ISO/IEC, 2009)

O objetivo principal do CC é ser a base para a avaliação das características de segurança dos sistemas de informação, permitindo comparações entre produtos similares, a partir de um conjunto de requisitos de segurança e um conjunto de medidas de segurança. Ao fim de uma avaliação entre dois produtos, são comparadas as funções de segurança e níveis de garantia, possibilitando a escolha do melhor aos negócios da organização. A Norma é composta por três volumes:

- **Volume 1 – Introdução e Modelo Geral**: Introdução a norma, definindo conceitos gerais e os princípios da avaliação da segurança em Tecnologia da Informação apresentando um modelo geral de avaliação.
- **Volume 2 – Componentes de Segurança Funcional**: Cria um conjunto de componentes de segurança funcional que servem como modelos de padrão para requisitos funcionais para os Alvos da Avaliação (Target Of Evalutation – TOE). Esse conjunto é organizado em famílias e classes.
- **Volume 3 – Componentes de Garantia de Segurança**: Cria um conjunto de componentes de garantia de segurança que servem como modelos de padrão para a garantia de requisitos básicos para os TOEs. Esse conjunto

é organizado em famílias e classes. Além disso, define critérios de avaliação para os *Protection Profile* (PP) e *Security Target* (ST) e apresenta sete pacotes pré-definidos de segurança que são chamados *Evaluation Assurance Levels* (EAL).

Essas normas foram concebidas para organizações que produzem e vendem software, não auxiliando o cliente durante a fase de avaliação dos softwares para futura aquisição. Para que isso ocorra é necessário criar critérios críticos para avaliação de acordo com o que o negócio necessita. Uma forma é realizar a comparação das características de segurança dos softwares a partir das métricas de segurança, principalmente pela natureza dinâmica do software. Henning (2001) realizou uma classificação em categorias para essas métricas:

- **Técnica**: Nesta categoria estão métricas para descrever e comparar, objetos técnicos como sistemas de algoritmos, especificações, arquiteturas e designs alternativos, produtos e sistemas implementados em diferentes fases do ciclo de vida do sistema.
- **Organizacional**: Nesta categoria estão métricas para descrever e acompanhar a eficácia de programas e processos organizacionais.
- **Operacional**: Nesta categoria estão métricas para descrever e gerenciar os riscos dos ambientes operacionais, incluindo práticas e sistemas.

Gollmann (2006) descreve que as métricas de segurança é o santo gral da engenharia de segurança. Para controlar melhor essas a engenharia de segurança pode ser dividida em: Criptografia de Dados, Identificação e Autenticidade, Auditabilidade e Log, Vulnerabilidades e Manipulação de Dados.

2.5.2 Criptografia de Dados

Criptografia de dados é o estudo de técnicas matemáticas relacionadas com o aspectos de segurança da informação como confidencialidade, integridade, autenticação de entidades e autenticação de dados de origem (MENEZES ET AL., 1996).

No passado, para cifrar mensagens haviam diversos processos criados pelos povos antigos que consistiam de duas etapas e um processo. A primeira etapa era o cifragem da mensagem pelo remetente utilizando um determinado processo. A segunda etapa consistia na decifragem da mensagem pelo destinatário utilizado o

mesmo processo de forma reversa. Portanto o remetente e o destinatário devem conhecer o processo a priori para transmitir a mensagem corretamente.

Atualmente a criptografia de dados é muito parecida. Os cifradores realizam o processo de cifragem e decifragem de uma informação. Qualquer pessoa pode criar um cifrador, mas Callas (2006) adverte, “é muito fácil criar um cifrador que seja bom o bastante para que você não o quebre, mas é muito difícil criar um cifrador que outras pessoas não possam quebrá-lo”.

Existem duas classes de cifradores, os assimétricos e os simétricos. Cifradores assimétricos possuem códigos de proteção muito maiores que os cifradores simétricos, mas são muito mais demorados da ordem de 10.000 vezes.

Ao definir o cifrador, o próximo passo a ser escolhido é a chave criptográfica que são os segredos da criptografia, sendo que a segurança da criptografia depende da forma que as chaves são criadas, usadas, protegidas e destruídas (CALLAS, 2006). Este define três formas principais de construir uma chave criptográfica:

1) Chaves Primárias: São cadeias de bits provenientes de um gerador de números aleatórios, sendo as mais comuns.

2) Chaves Derivadas: São produzidas a partir de algum item, como senhas.

3) Chaves Estruturadas: É um tipo de chave derivada produzida a partir de alguns números aleatórios.

Existem alguns fatores relacionados com as chaves criptográficas que devem ser levados em consideração para a escolha do melhor cifrador para o componente. Esses fatores são descritos por Perlner e Cooper (2009):

- **Tamanho das chaves, mensagens de troca de chaves e assinaturas**: Grande parte dos algoritmos possui o mesmo tamanho para esses itens, variando entre centenas e milhares de bits. Caso o tamanho seja muito grande pode haver gargalos na transmissão devido à largura da banda ou dispositivos com memória limitada não suportando tamanhos maiores.
- **Vida útil da chave criptográfica:** Durante a transcrição de mensagens assinadas, estas podem revelar informações sobre a chave privada do assinante, limitando o número de mensagens que podem ser seguramente assinadas com a mesma chave. O exemplo mais extremo está no esquema Lamport (1979), que requer uma nova chave para nova cada mensagem. Um detalhe é que as chaves privadas usadas para a

decifragem geralmente não possuem tempo de vida limitado, pois a cifragem não a utiliza e os decodificadores não revelam informações.

- **Custo Computacional:** Existem quatro operações básicas para chaves criptográficas: Codificação, Decodificação, Assinaturas e Verificação de Assinaturas. Atualmente essas têm um custo de alguns milissegundos, exceto para Codificação RSA e Verificação de Assinatura, que podem ser cerca de cem vezes mais rápido devido ao uso de expoentes públicos. O tempo de geração de uma chave pública pode preocupar, pois é muito mais caro que as operações básicas. Os algoritmos baseados em fatoriais tendem a ter esse problema, pois a geração dos fatores primos com uma alta entropia requer alguns segundos de computação.

As chaves criptográficas são geradas pelos cifradores de acordo com o tipo do algoritmo, sendo ou simétricos ou assimétricos. Para maiores detalhes sobre os itens internos dessa característica, favor consultar o APÊNDICE A – Tópicos de Criptografia de Dados.

2.5.3 Identificação e Autenticidade

Identificação ou autenticação de entidade é o processo onde uma parte é assegurada, via aquisição de alguma evidência corroborativa, da identidade de uma segunda parte envolvida no protocolo, e que a segunda parte está atualmente participando, isto é, é ativa no momento em que a evidência foi adquirida (MENEZES ET AL., 1996).

Dessa forma o objetivo de uma autenticação é que a parte que está clamando possua uma identificação, barrando possíveis ataques caso não a tenha. Esta é intransferível, ou seja, somente a parte A pode realizar a autenticação pela parte A. As técnicas de autenticação são divididas em três categorias principais:

- **Algo Conhecido**: Algo que o individuo conhece e que pode ser utilizado para realizar o ingresso em um sistema, como por exemplo, senhas, PINs (*Personal Identification Numbers*) ou desafios e respostas sobre determinada informação privada.
- **Algo de posse**: Algo que o individuo possui, normalmente um acessório físico com função semelhante à de um passaporte. Exemplos são os cartões magnéticos, cartões com chip e tokens.

- **Algo inerente**: Algo que o individuo possui de intrínseco fazendo com que seja único. Essa categoria inclui métodos que utilizam de características físicas como impressão digital, iris, voz ou características involuntárias como assinaturas, forma de escrita entre outros.

Estas classes são formadas por alguns tipos de identificação. Estes itens serão descritos de maneira mais detalhada no APÊNDICE B – Tópicos de Identificação e Autenticidade.

2.5.4 Auditabilidade e Log

Kent e Souppaya (2006) definem log como uma forma de gravar eventos que ocorrem nos sistemas e redes de uma organização. Estes são compostos por itens chamados de entradas de log, onde estas são informações relacionadas com eventos que ocorreram na organização, normalmente relacionados à segurança.

Existem duas formas de um componente gerar um arquivo de log. A primeira é a criação de seus próprios arquivos, sendo a forma mais comum. A segunda é a utilização do sistema de log do projeto ou o software de log do sistema operacional.

Nos primórdios, os arquivos log armazenavam somente os problemas de um sistema. Atualmente eles são utilizados para atividades como aperfeiçoar o desempenho de sistemas e da rede de comunicação, gravar a ação de usuários e prover dados úteis para investigar atividades maliciosas. Para a realização dessas atividades é necessário ocorrer operações sobre os arquivos de log. Estes itens serão descritos de maneira mais detalhada no APÊNDICE C – Tópicos de Auditabilidade e Log.

2.5.5 Ameaças e Vulnerabilidades

Vulnerabilidade é uma instância de uma falta na especificação, desenvolvimento ou configuração de um software, tal que sua execução pode violar uma política de segurança implícita ou explícita do sistema (SHIN; WILLIAMS, 2008). Uma vulnerabilidade pode ser classificada em três classes principais (AVIZIENIS ET AL., 2004):

- **Falha**: Evento que ocorre quando a funcionalidade sofre um desvio do seu funcionamento correto. Pode estar relacionado a um problema na especificação funcional do sistema.

- **Erro**: É a causa de uma falha, ou seja, do desvio do funcionamento correto do sistema.
- **Falta**: É a declaração ou hipótese de um erro. Pode ser ativa, quando causa o erro ou inativa, quando não o causa. E pode ser interna ou externa ao sistema.

Estes itens serão descritos de maneira mais detalhada no APÊNDICE D – Tópicos em Ameaças e Vulnerabilidades.

2.5.6 Manipulação de Dados de Entrada

Manipulação de dados de entrada é o processo de validação de dados fornecido por outras entidades em um conjunto de regras pré-definidas (NETLAND ET AL., 2006). OWASP (2012) define três categorias:

- **Checagem de Integridade**: Certificar que os dados de entrada não foram adulterados.
- **Validação**: Realizar uma validação correta de acordo com o dado de entrada. Algumas dessas validações são:
  - Certificar que o dado é fortemente tipado
  - Certificar que o dado possui uma sintaxe correta
  - Certificar que o dado respeita os limites de tamanho
  - Certificar que o dado possui somente caracteres permitidos
  - Se o dado for um número certificar que estão dentro dos limites e sinais corretos.
- **Regras de Negócio**: Certificar que os dados de entrada não estão somente validados, mas com regras de negócio corretas.

O objetivo principal é validar os dados de entrada, não tornando as soluções de segurança de rede obsoletas. De acordo com Netland et al. (2006) essa “validação é um mecanismo de defesa suplementar de ajuda para deixar o ambiente seguro para a execução do sistema”. Dessa maneira, em uma aplicação web, quaisquer chamadas de verificações do lado do cliente são inúteis, pois pode ocorrer um ataque de interceptação de informações durante a transmissão para o servidor.

Uma validação extensiva sobre os dados de entrada significa mais sobrecarga computacional, assim os desenvolvedores devem se esforçar para validar de maneira menos intrusiva possível, principalmente em sistemas de dados

intensivos, onde atrasos no processamento podem deixar o sistema inútil. Os itens que compõem essa característica serão descritos de maneira mais detalhada no APÊNDICE E – Tópicos em Manipulação de Dados de Entrada.

2.6 Aspectos de Segurança em Componentes de Software

Um sistema desenvolvido sobre o CBD pode conter um grande número de componentes. Dependendo do tamanho do sistema, esse número pode chegar à casa de centenas de componentes. Segundo Khan, Han e Zheng (2000):

> "Em um sistema baseado em componentes de software, cada componente tem suas propriedades próprias de segurança, e estas precisam ser devidamente especificadas, de tal forma que os outros componentes, bem como os usuários, possam ser capazes de "entender" esses recursos e o seu impacto ao realizar a inserção no sistema. (p. 58, tradução nossa)

Assim, todo componente deve possuir uma documentação clara sobre as características de segurança. Um conjunto mínimo de características, principalmente para dispositivos móveis, é descrito por Campadello, Maclaverty e Saridakis (2004):

- **Autenticação**: Todo ator deve ser autenticado. Somente se todos mantiverem sua identificação pode ser estabelecido o relacionamento entre os dispositivos de usuário e os provedores de serviços.
- **Integridade**: Durante o ciclo de vida, o componente deve manter sua integridade. Esta deve ser protegida de tentativas de adulteração. Um componente adulterado pode ser usado para enfraquecer a funcionalidade de todo o sistema, para coletar informações confidenciais ou para produzir resultados com problemas.
- **Confidencialidade:** A informação que estiver armazenada dentro do componente e o próprio componente devem permanecer confidenciais a entidades terceiras. O componente deve ser mantido de forma confidencial pelo provedor para evitar duplicações não autorizadas, enquanto seu conteúdo pode conter informações confidenciais.

Uma vertente da área de segurança encontra-se no monitoramento de componentes. Lenzini, Tokmakoff e Muskens (2007) definem um framework de métricas para avaliação em tempo real de componentes. Esse, dependendo das medidas coletadas, pode reiniciar ou parar um componente que esteja sendo monitorado. O nome desse framework é *Trustworthiness Management Framework* e

é uma evolução do modelo proposto pelo projeto *Robocop/Space4U* do ITEA (*Information Technology for European Advancement*) pertencente ao projeto *Trust4All* encerrado em 2007. O framework é dividido em duas partes principais: o *Trustworthiness Model*, que define um modelo de qualidade, com modos, atributos de qualidade e métricas de qualidade; e o *Trust Management Framework Model*, que é o modelo que irá monitorar as métricas.

Outra vertente está relacionada com a certificação de características de segurança nos componentes. Um dos primeiros trabalhos nessa área é o de Ghosh e McGraw (1998), que retrata uma certificação de segurança em componentes. Eles dividem a certificação em duas partes: a certificação em nível de componente e a em nível de sistema. Além disso, ele apresenta algumas formas de certificação em Caixa Branca e em Caixa Preta.

Nessa mesma época, iniciou-se a apresentação de uma série de trabalhos de Khan, Han (2000), (2002), (2003), (2004), (2005), (2006), (2008). Este cria um framework de processos para caracterização de segurança em componentes de software e suas composições, ou seja, a integração entre eles. O framework possui quatro fases. A primeira fase é a caracterização de propriedades de segurança de componentes atômicos. A segunda é a de aprovação das propriedades de segurança. A terceira é a composição de contratos de segurança. A quarta fase é a composição de contratos de segurança do sistema.

A primeira e a segunda fase realizam uma validação sobre um componente em sua atomicidade. Durante a primeira fase, são identificadas as funcionalidades existentes e suas interfaces, que acessam as funções do componente. Após a identificação dessas interfaces, o framework levanta quais possuem funções de segurança, como itens de autenticação ou confidencialidade e realiza a extração das características de segurança armazenando em um arquivo XML. Ao levantar as características, o framework leva em conta as propriedades definidas dentro da Norma 15408, a *Common Criteria* (ISO/IEC, 2009).

A segunda fase consiste em quatro passos. O primeiro verifica a consistência entre o documento de especificação de interfaces e as interfaces construídas. O segundo é a aprovação das interfaces do componente de acordo com selos de certificação digital. O terceiro passo é a criação de lacres para a interface aprovada e o último passo é o registro do certificado dentro da versão do componente.

Este framework realiza um estudo direcionado para características de segurança de um componente, mas não cria um modelo de características e sim uma documentação durante a fase de desenvolvimento do componente. Dessa forma não é realizada uma modelagem de propriedades que podem existir, pois estas estão fortemente ligadas com a norma da Common Criteria.

Outro ponto que diverge é que como é um framework de desenvolvimento de componentes seguindo uma política para especificação de características de segurança, não há a possibilidade de avaliar um componente existente que não seguiu esse framework, sem criar uma nova documentação de acordo com a proposta pelo framework.

2.7 Considerações sobre o capítulo

Este capítulo apresentou os principais conceitos aplicados ao longo desse trabalho como reúso de software, componentes de software, modelos de qualidade e áreas de segurança. A questão principal envolvida é se as características internas de segurança de um componente de software possuem uma qualidade aderente aos padrões esperados pelo projeto. Foram revisados nesse capitulo diversas áreas de segurança como Criptografia de Dados, Autenticidade e Identificação, Auditabilidade e Log, Vulnerabilidades e Ameaças e Validação de Dados de Entrada. Esses são conceitos fundamentais para dar forma a um modelo de qualidade que defina quais são as características e subcaracterísticas importantes dentro da área de segurança.

CAPÍTULO 3 - ESTRUTURAÇÃO DA PESQUISA

Algumas pessoas acham que foco significa dizer sim para a coisa em que você vai se focar. Mas não é nada disso. Significa dizer não às centenas de outras boas ideias que existem. Você precisa selecionar cuidadosamente.
- Steve Jobs – Criador da Apple

O objetivo deste capítulo é descrever os métodos de pesquisa utilizados neste trabalho, detalhando as estratégias adotadas para atingir o objetivo geral.

3.1 Conceitos relevantes sobre metodologia e métodos de pesquisa

Uma pesquisa é “um procedimento racional e sistemático que tem como objetivo proporcionar respostas aos problemas que são propostos” (GIL, 2002). O autor ainda define que um problema é “uma questão não resolvida que é objeto de discussão, em qualquer domínio do conhecimento”. Assim uma pesquisa deve procurar resolver um problema. De acordo com Gil (2002), existem três categorias de pesquisa:

- **Pesquisa Exploratória:** O objetivo é proporcionar maior familiaridade com o problema, tornando-o mais explícito ou construindo hipóteses, aprimorando as ideias ou descobrindo intuições. Possui um planejamento flexível, considerando os mais variados aspectos do fato estudado. Em sua maioria, essas pesquisas envolvem: a) levantamento bibliográfico; b) entrevistas com pessoas quer possuem experiências práticas com o problema; e, c) análise de exemplos para estimular a compreensão.
- **Pesquisa descritiva:** O objetivo primordial é a descrição das características de determinada população ou fenômeno ou o estabelecimento de relações entre variáveis. São inúmeros os estudos que podem ser classificados sob esta ótica e uma de suas características mais significativas é a utilização de técnicas padronizadas de coleta de dados, tais como o questionário e a observação sistemática.

- **Pesquisas explicativas:** O objetivo é identificar os fatores que determinam/contribuem para a ocorrência dos fenômenos. Este tipo de pesquisa é a que mais aprofunda o conhecimento da realidade, porque explica a razão, o porquê das coisas. Por isso, é o tipo mais complexo e delicado, já que o risco de cometer erros aumenta consideravelmente. Pode-se dizer que o conhecimento científico está assentado nos resultados oferecidos pelos estudos explicativos.

Outra categorização existente é quanto aos procedimentos técnicos utilizados para realizar as pesquisa. Gil (2002) divide esses procedimentos em dois grupos: *Fontes de Papel* e *Dados Fornecidos por Pessoas*. No primeiro grupo, existe uma divisão entre Pesquisa Bibliográfica e Pesquisa Documental:

- **Pesquisa Bibliográfica:** Desenvolvida com base em materiais já elaborados, sendo constituído principalmente de livros de referência e artigos científicos.
- **Pesquisa Documental:** Parecida com a Pesquisa Bibliográfica, mas com materiais que ainda não receberam um tratamento analítico.

No segundo grupo, Dados Fornecidos por Pessoas, existe uma subdivisão com os seguintes subgrupos:

- **Pesquisa Experimental:** Consiste em determinar um objeto de estudo, selecionar as variáveis capazes de influenciá-lo, definir as formas de controle e observação dos efeitos dessas variáveis sobre o objeto.
- **Pesquisa Ex-post Facto**: (Pesquisa a partir do fato passado) É um estudo realizado após ocorrência de variações na variável dependente no curso natural dos acontecimentos.
- **Pesquisa Estudo de Corte:** Se refere a um grupo de pessoas que tem alguma característica em comum, constituindo uma amostra a ser acompanhada por período de tempo em relação ao fato investigado.
- **Pesquisa de Levantamento:** Caracteriza-se pela interrogação direta das pessoas cujo comportamento se deseja conhecer.
- **Pesquisa de Estudo de Campo:** Semelhante ao levantamento, porém com maior profundidade. Utilizam-se mais técnicas de observação do que de interrogação. Tipicamente focaliza uma comunidade. O pesquisador realiza a maior parte do trabalho pessoalmente.

- **Pesquisa de Estudo de Caso:** Refere-se ao estudo profundo e exaustivo de um ou poucos objetos, permitindo detalhar o conhecimento.
- **Pesquisa-ação:** Possui o envolvimento ativo do pesquisador e ação por parte das pessoas ou grupos envolvidos no problema.
- **Pesquisa Participante:** Caracteriza-se pela interação entre pesquisadores na qual o pesquisador toma parte da ação.

3.2 Caracterização da pesquisa

De acordo com os objetivos desse trabalho, esse foi caracterizado como uma Pesquisa Exploratória, pois pretende explorar as características de segurança em componentes de software visando à criação de um novo modelo de qualidade voltado para as características relevantes de segurança sobre componentes.

3.3 Estratégia de pesquisa

A partir desse tópico serão explanadas as etapas que formam esse trabalho. Assim, de acordo com a Figura 3-1 o estudo foi desenvolvido em quatro etapas que serão detalhadas a seguir.

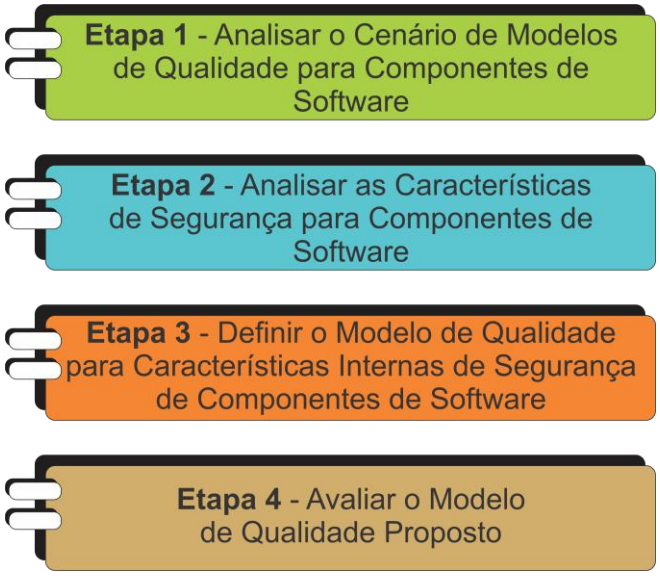

Figura 3-1. Metodologia do Trabalho (Fonte: O Autor).

3.3.1 Analisar o Cenário de Modelos de Qualidade para Componentes de Software

A primeira etapa teve o objetivo de pesquisar na literatura trabalhos acadêmicos que propuseram novos modelos de qualidade para componentes de

software. Esses foram utilizados para criar a base de conhecimento para o desenvolvimento de um novo modelo nos padrões da engenharia de software.

Para atingir o objetivo foi utilizada a revisão sistemática. Esta revisão consiste em identificar, avaliar e interpretar as pesquisas disponíveis e relevantes para uma determinada questão (KITCHENHAM ET AL., 2004). Para realizar essa revisão foram executadas as seguintes fases:

- Planejar o Processo e o Objetivo da Revisão Sistemática
- Identificar as Palavras Chaves
- Identificar as Bases de Estudo
- Selecionar Buscas e Estudos Primários
- Identificar Estudos Brasileiros

3.3.1.1 Planejar o Processo e o Objetivo da Revisão Sistemática

Nesta fase foi estabelecido o processo de aplicação da revisão sistemática. Esse consiste na definição do objetivo, das questões de pesquisa, das fontes primárias de estudo e dos critérios para inclusão e exclusão de artigos. A busca dos modelos de qualidade para componentes de software foi o objetivo desta revisão. As questões que basearam essa busca foram:

(I) Quais são os modelos de qualidade para componentes de software existentes?

(II) Quais as características mais utilizadas nos modelos de qualidade para componentes de software existentes?

(III) Quais as subcaracterísticas mais utilizadas nos modelos de qualidade para componentes de software existentes?

3.3.1.2 Identificar as Palavras Chaves

O principal objetivo de uma revisão sistemática é encontrar o maior número de estudos possível dentro do tópico relacionado, utilizando uma estratégia imparcial de busca (KITCHENHAM ET AL., 2004). Assim são combinadas palavras chaves para buscar estes artigos. As palavras chaves utilizadas foram:

(I) “Software Component Model”

(II) “Component Quality Model”

(III) “Software Quality Model” AND Component

Esses três conjuntos de palavras abrangem a taxonomia mais utilizada na definição dos modelos de qualidade, não sendo necessário incluir as palavras *IT* ou *software engineering*. A busca foi ampliada para abranger todo o conteúdo do estudo, não se limitando apenas ao titulo, *abstract* e palavras chave. Além disso, foram combinados os três grupos de palavras chave em novas buscas, mas a quantidade de estudos retornados permaneceu igual.

3.3.1.3 Identificar as Bases de Estudo

A definição das bases de pesquisa para a busca dos estudos é um ponto muito importante da revisão. Estas bases foram escolhidas por incluírem os periódicos e anais de conferências relevantes para a área de Ciência da Computação, abrangendo várias áreas do conhecimento dentro da engenharia de software. Assim foram definidas quatro bases:

- *ACM Digital Library (http://portal.acm.org)*
- *IEEE Xplore Digital Library (http://www.ieee.org/web/publications/xplore/)*
- *ScienceDirect – Elsevier (http://www.elsevier.com)*
- *SpringerLink (http://www.springerlink.com)*

3.3.1.4 Selecionar Buscas e Estudos Primários

O último ponto de critério de seleção dos estudos foi a limitação do ano de publicação do artigo, limitando-se aos últimos 10 anos, ou seja, somente foram incluídos estudos a partir do ano de 2002. Essa decisão ocorreu a partir do fato que a última versão da norma ISO/IEC 9126 (ISO/IEC, 2001) foi lançada em 2001, assim ao escolher o ano de 2002 fica indicado que somente os últimos estudos relacionados a modelos de qualidade foram retornados pela busca. As buscas foram realizadas no mês de janeiro de 2011 retornando um total de 433 estudos.

De acordo com Kitchenham et al. (2004), o primeiro passo após a busca consiste na eliminação dos duplicados. De 433 estudos, foram encontrados 58 estudos duplicados, assim o número total de estudos foi reduzido para 375.

A segunda parte da exclusão foi realizada com a leitura de todos os artigos, não somente o título, *abstract* e palavras chaves. Essa leitura verificou o assunto do artigo buscando modelos de qualidade inéditos, ou seja, que não descrevessem ou reproduzissem modelos já existentes na literatura.

Assim foram excluídos estudos que não tivessem como tema a criação de um novo modelo de qualidade. Alguns estudos especializavam algumas características, como a usabilidade, mas era uma descrição mais especializada, acrescentando muitas vezes novas subcaracterísticas, atributos ou métricas, tornando um pequeno pedaço mais especializado. Esses estudos foram excluídos, pois não realizavam a descrição de um modelo único, mas apenas de uma parte. Com esses critérios de exclusão restaram sete estudos, que foram analisados com uma maior profundidade.

Após a conclusão da revisão da literatura em fevereiro de 2012, foram realizadas novas buscas com os mesmos termos chaves, mas não apresentaram novos resultados.

Foi realizada uma busca via *Google Scholar* e *Microsoft Academic* de artigos brasileiros, já que não existe uma base de indexação única para a pesquisa de material acadêmico nacional. Dessa busca foram obtidos dois modelos de qualidade que não haviam sido retornados na revisão sistemática.

Todos os modelos retornados estão descritos no capítulo 2, com uma análise sobre suas características e subcaracterísticas. A análise dos itens de segurança nesses modelos será apresentada no capitulo 4.

3.3.2 Analisar as Características de Segurança para Componentes de Software

O foco desta etapa foi a busca de características de segurança para componentes de software, com duas perspectivas principais: a busca de características de segurança a partir dos modelos de qualidade e busca de características de segurança tendo como foco principal a área de segurança.

3.3.2.1 Buscar Características nos Modelos de Qualidade

Consiste na definição de características de segurança para componentes a partir dos modelos de qualidade obtidos na etapa anterior. É necessário analisar as características listadas pelos autores dos modelos, avaliando dois pontos principais. O primeiro é referente à presença das características de segurança e o segundo ponto é referente à profundidade da descrição destas características.

Assim no primeiro ponto foram avaliados os modelos de qualidade quanto à presença das características de segurança. Caso houvesse essa presença, foi

realizado o segundo ponto, que consistia na análise de sua importância sobre o modelo. Essa análise consistiu na avaliação das descrições do modelo sobre as subcaracterísticas, atributos e métricas auxiliando o desenvolvedor a avaliar se o seu componente possui um bom padrão de qualidade.

3.3.2.2 Buscar Características na Área de Segurança

Consiste na busca e análise de características na área de segurança, obtendo um conhecimento mais profundo sobre essas áreas dentro da Tecnologia da Informação. Para essa busca foram consultados artigos e livros de segurança da informação (MENEZES ET AL., 1996), (HOWARD; LEBLANC, 2002), (AVIZIENIS ET AL., 2004), (WINDLEY, 2005), (CALLAS, 2006), (GOLLMANN, 2006), (KENT; SOUPPAYA, 2006), (PERLNER; COOPER, 2009) e (JACOBS; POLL; 2011) obtendo as principais áreas. Essas áreas foram validadas com um especialista em segurança de tecnologia da informação, que indicou os melhores materiais para buscar uma informação mais específica para esses itens.

Ao término da leitura dos materiais repassados pelo especialista foram separados os de maior importância e contextualizados com o escopo de componentes, constituindo assim as características do novo modelo de qualidade. Esses tópicos reservados da área de segurança por essa etapa foram: Criptografia de Dados, Autenticidade e Identificação, Auditabilidade e Log, Ameaças e Vulnerabilidades e Manipulação de Dados.

3.3.3 Definir o Modelo de Qualidade para Características Internas de Segurança de Componentes de Software

Essa etapa define o modelo de qualidade para características internas de segurança para componentes de software. Para isso foram realizadas três tarefas em paralelo: Definição de Características, Definição de Subcaracterísticas, Definição de Atributos e Tipos.

3.3.3.1 Definir Características

Consiste em definir e analisar cada uma das características selecionadas na etapa 2. Essa análise buscou materiais especializados para cada característica. Essa busca foi realizada sobre as bases de estudo identificadas na primeira revisão

sistemática. Com esse material, inicia-se a etapa de definição de subcaracterísticas e definição de atributos e tipos em paralelo, pois ao realizar a leitura de artigos e livros de uma determinada área, já foram elencados os itens internos à característica.

3.3.3.2 Definir Subcaracterísticas

Consiste em obter uma lista de subcaracterísticas importantes dentro de cada característica selecionada. Dessa forma foram definidas 14 subcaracterísticas importantes dentro das cinco características selecionadas. A partir dessas subcaracterísticas podem ser agrupados os atributos que as compõem.

3.3.3.3 Definir Atributos e Tipos

Consiste em obter uma lista de atributos, tipos e exemplos que são importantes dentro de cada subcaracterística selecionada. Assim foram definidos 44 atributos junto com quatro formas de classificação (tipos). Esses atributos consistem no item mais básico do modelo, podendo o desenvolvedor utilizar para listar possíveis candidatos a melhoramento da segurança em um componente de software.

3.3.4 Avaliar o Modelo de Qualidade Proposto

Essa última etapa consistiu na avaliação do modelo de qualidade proposto por esse trabalho. Garcia (2010) descreve que a avaliação de um modelo de qualidade não pode ser realizada por experimentos controlados, pois todo o cenário de desenvolvimento deve ser analisado por um longo período de tempo, tendo como consequências que as variáveis independentes não podem ser controladas e a análise dos resultados é muito difícil de ser obtida. Assim é necessário obter uma forma de avaliar um modelo de qualidade e uma dessas formas é a avaliação conduzida por especialistas. Garcia (2010) faz uma definição sobre essa avaliação:

> “Uma avaliação conduzida por especialistas é uma série de esforços científicos que são empregados para interpretar dados, predizer o comportamento de um determinado sistema e avaliar possíveis incertezas. Para isso são utilizadas especulações, palpites e estimativas pelas pessoas que são consideradas especialistas, servindo de entradas cognitivas em alguns processos de decisão.” (p. 76, tradução nossa)

Assim a avaliação do modelo deste trabalho consistiu em uma pesquisa aplicada por meio de um questionário contendo perguntas divididas em três grupos, conforme apresentado no Apêndice A:

(I) Caracterização do profissional;

(II) Caracterização das áreas de segurança para produtos de software; e,

(III) Avaliação do modelo de qualidade proposto propriamente dito.

3.4 Considerações sobre o capítulo

Este capítulo apresentou os conceitos relevantes à estruturação da pesquisa científica, estabelecendo uma classificação inicial dos tipos de pesquisas a serem abordados neste trabalho. As etapas da pesquisa foram detalhadas de forma a esclarecer e delimitar os procedimentos a serem executados ao longo da execução deste trabalho. As principais etapas foram:

- Analisar o Cenário de Modelos de Qualidade para Componentes de Software
- Analisar as Características de Segurança para Componentes de Software
- Definir o Modelo de Qualidade para Características Internas de Segurança de Componentes de Software
- Avaliar o Modelo de Qualidade Proposto

Os resultados obtidos nestas etapas serão explicados no próximo capitulo.

CAPÍTULO 4 - DESENVOLVIMENTO DA PESQUISA

Acrescentemos alguma coisa a uma qualidade, e ela parecerá defeito. Tiremos alguma coisa de um defeito, e ele parecerá qualidade.
- Émile Faguet – Escritor Frances e critico literário.

Este capítulo apresenta o modelo de qualidade de características internas de segurança de componentes de software, que foi desenvolvido de acordo com os passos descritos na metodologia de trabalho definida no capitulo 3. Esses passos consistem nos seguintes itens descritos no decorrer deste capitulo:

- Obtenção, análise e comparação dos modelos de qualidade para componentes de software
- Definição das características de segurança para componentes de software
- Definição do modelo de qualidade de características internas de segurança de componentes de software
- Avaliação do modelo de qualidade proposto por especialistas

4.1 Analisar o Cenário de Modelos de Qualidade para Componentes de Software

Para que um componente de software atenda à expectativa do usuário é necessário que este tenha qualidade. O IEEE (1990) define qualidade de software como sendo “o grau que um sistema, componente de sistema ou processo, atende a determinados requisitos ou a necessidade ou a expectativa do cliente ou usuário”.

Garvin (1984) segue esse mesmo contexto da definição, mas no contexto de qualidade como um todo a partir de visões. Essas visões podem ser aproveitadas na definição dessa qualidade a ser atingida. O modelo definido neste trabalho irá utilizar a visão do usuário, a visão do produtor e no que for possível a visão do produto. Desse embasamento na qualidade surge uma primeira questão “*O que é necessário*

enumerar dentro de um componente para que a sua qualidade esteja de acordo com a expectativa do usuário?".

Com essa questão é iniciada a tarefa de buscar nas bases de estudo se existiam os modelos de qualidade para componentes e quais eram as definições destes. Mas essa busca não poderia ser desordenada, pois deve gerar a possibilidade de ser refeita caso necessário. Assim uma busca com essa característica é a Revisão Sistemática, que consiste em um processo formalizado e repetível para documentar um conhecimento de uma determinada área, seguindo uma série de passos pré-estabelecidos (KITCHENHAM ET AL., 2004). O processo e os itens necessários para realizar essa revisão foram descritos com maiores detalhes na seção 3.3.1.

O importante aqui é destacar que foram obtidos sete estudos que definem modelos de qualidade para componentes de software. Esses estudos foram definidos com maiores detalhes na seção 2.4 e são os seguintes:

- Modelo de Bertoa e Vallecillo (2002)
- Modelo de Rawashdeh e Matalkah (2006)
- Modelo de Andreou e Tziakouris (2007)
- Modelo de Choi et al. (2008)
- Modelo de Sharma, Kumar e Grover (2008)
- Modelo de Alvaro (2009)
- Modelo de Kalaimagal e Srinivasan (2010),

Como descrito nos detalhes da revisão sistemática na seção 3.1.1, não existe uma base de indexação para os estudos científicos brasileiros, assim foi necessária uma busca utilizando as mesmas palavras chaves, mas em português no *Google Scholar* e *Microsoft Academic*, resultando em dois estudos que foram definidos com maiores detalhes na seção 2.4 e são os seguintes:

- Modelo de Simão e Belchior (2002)
- Modelo de Colombo et al. (2007)

Pelo planejamento, a revisão sistemática buscou apenas os modelos definidos a partir de 2002 pelo motivo da data de publicação da ultima versão da norma ISO/IEC 9126, e em sua maioria baseia-se no padrão ISO/IEC 9126 (ISO/IEC, 2001), apenas o modelo de Kalaimagal e Srinivasan (2010) foi baseado no padrão ISO/IEC 25000 (ISO/IEC, 2005). Dessa forma, todos utilizam o padrão

hierárquico definido pelos modelos dos padrões ISO/IEC, com características e subcaracterísticas e alguns apresentam ainda atributos e métricas.

As métricas foram definidas de uma forma muito generalizada, não auxiliando o desenvolvedor a definir passos para a obtenção de valores condizentes para uma boa análise. Outro ponto é relacionado com a superficialidade dos itens que constituem os modelos, pois estes não se aprofundam em áreas específicas como a segurança da informação.

4.2 Analisar as Características de Segurança para Componentes de Software

Essa etapa foi responsável por aprofundar os estudos sobre os modelos de qualidade encontrados na etapa anterior. Esse aprofundamento consistiu em observar os itens dos modelos de qualidade (características, subcaracterísticas, atributos) e verificar se esses itens continham valores referentes à área de segurança e como estes eram tratados pelo modelo.

A partir dos conceitos obtidos por esse aprofundamento, se inicia uma subetapa de aprofundamento desses conceitos a partir da busca de estudos dentro da área de segurança de software.

4.2.1 Características de Segurança Baseadas nos Modelos de Qualidade

O objetivo dessa subetapa é analisar os modelos encontrados na etapa anterior, junto com os modelos dos padrões da ISO/IEC a fim de encontrar itens relacionados à segurança de componentes de software. Essa análise constitui-se dos seguintes passos:

(I) Analisaram-se as características de todos os modelos buscando o relacionamento com a área de segurança

(II) Analisaram-se as subcaracterísticas de todos os modelos buscando o relacionamento com a área de segurança

(III) Analisaram-se os atributos e possíveis métricas de todos os modelos buscando o relacionamento com a área de segurança.

A maioria dos modelos de qualidade apresentou no passo I e II uma subcaracterística chamada de *Segurança* como um subitem da característica de *Funcionalidade*. Os modelos que definiram essa variável foram os seguintes:

- Modelo do padrão ISO/IEC 9126 (2001)
- Modelo de Bertoa e Vallecillo (2002)
- Modelo de Simão e Belchior (2002)
- Modelo de Rawashdeh e Matalkah (2006)
- Modelo de Andreou e Tziakouris (2007)
- Modelo de Colombo et al. (2007)
- Modelo de Choi et al. (2008)
- Modelo de Sharma, Kumar e Grover (2008)
- Modelo de Alvaro (2009)

O modelo de qualidade do padrão ISO/IEC 25010 (ISO/IEC, 2005) definiu uma característica chamada de *Segurança* com cinco subcaracterísticas, *Confidencialidade*, *Integridade*, *Não Repúdio*, *Responsabilização* e *Autenticidade*. As definições dessas subcaracterísticas serão apresentadas mais adiante, na descrição do item de atributos dos modelos de qualidade.

O modelo de Kalaimagal e Srinivasan (2010) cria duas características para a área de segurança, a característica *Segurança* e a característica *Segurança em Uso*. A primeira possui duas subcaracterísticas, *Autenticação* e *Criptografia de Dados*. A segunda possui quatro subcaracterísticas, *Risco do Software*, *Riscos Comerciais em Uso*, *Risco do Operador em Uso* e *Risco de Uso Público*. Todas as subcaracterísticas serão definidas mais adiante, na descrição do item de atributos dos modelos de qualidade. Para finalizar os passos I e II dessa subetapa, o Quadro 4-1 apresenta um resumo dos modelos em contra partida com as características e subcaracterísticas.

O passo III consiste na análise dos atributos de segurança dos modelos de qualidade. Para melhorar a análise foi realizado um nivelamento dos atributos em todos os modelos. Esse nivelamento consistiu em transformar subcaracterísticas presentes no modelo do padrão ISO/IEC 25010 (2005) e no modelo de Kalaimagal e Srinivasan (2010) em atributos para realizar a comparação com os atributos dos outros modelos de qualidade.

Quadro 4-1. Quadro Comparativo de Características e Subcaracterísticas entre os Modelos de Qualidade

Características	Subcaracterísticas	ISO/IEC 9126 (2001)	ISO/IEC 25010 (2005)	Bertoa e Vallecillo (2002)	Simão e Belchior (2002)	Rawashdeh e Matalkah (2006)	Andreou e Tziakouris (2007)	Colombo et al. (2007)	Choi et al. (2008)	Sharma, Kumar e Grover(2008)	Alvaro (2009)	Kalaimagal e Srinivasan (2010)
Funcionalidade	Segurança	X		X	X	X	X	X	X	X	X	
Segurança	Confidencialidade		X									
Segurança	Integridade		X									
Segurança	Não-Repúdio		X									
Segurança	Responsabilização		X									
Segurança	Autenticidade		X									
Segurança	Autenticação	X										X
Segurança	Criptografia de Dados											X
Segurança em Uso	Risco do Software											X
Segurança em Uso	Riscos Comerciais em Uso											X
Segurança em Uso	Risco Operador em Uso											X
Segurança em Uso	Risco de Uso Público											X

Desse nivelamento quatro modelos (ISO/IEC 9126 (2001), Rawashdeh e Matalkah (2006), Choi et al. (2008) e Sharma, Kumar e Grover (2008)) não apresentaram atributos em seus estudos públicos. No total foram citados quinze atributos nos modelos de qualidade. A definição de cada um desses é a seguinte:

- ***Auditabilidade:*** Atributo responsável por indicar se o componente possui algum mecanismo de auditoria, com capacidades de gravar ações de usuários sobre o sistema e seus dados (Bertoa e Vallecillo (2002)).
- **Autenticidade**: Atributo responsável pela identificação de recursos, verificando se a identificação deste é correta, sendo ele o dono da identificação. (ISO/IEC 25010 (2005)).
- **Confidencialidade**: Atributo responsável pelo o grau de proteção contra divulgação não autorizada de informações, independente de esta divulgação ser acidental ou deliberada (ISO/IEC 25010 (2005)).
- ***Autenticação:*** Atributo responsável por definir interfaces para o acesso às funcionalidades do componente (Kalaimagal e Srinivasan (2010)).

- **Criptografia de Dados:** Atributo responsável por proteger os dados contidos dentro do componente ou obtidos a partir deste (Kalaimagal e Srinivasan (2010)).
- **Integridade**: Atributo responsável pela identificação de alterações dentro de um dado do componente, alertando assim que ocorreu uma modificação não autorizada a este dado (ISO/IEC 25010 (2005)).
- **Não-Repúdio**: Atributo responsável por provar que um determinado usuário realizou determinada ação e que não pode ser negada futuramente (ISO/IEC 25010 (2005)).
- **Privacidade**: Atributo responsável por verificar se o componente possui mecanismos que garantam a privacidade dos usuários, com relação à utilização das informações (Simão e Belchior (2002)).
- **Resistencia a Escalabilidade de Privilégios**: Atributo responsável por verificar se uma falha de segurança do componente pode resultar em um aumento de privilégios de um usuário, causando uma violação de segurança (Andreou e Tziakouris (2007)).
- **Responsabilização**: Atributo responsável por infligir ações para uma entidade e somente para ela (ISO/IEC 25010 (2005)).
- ***Riscos Comerciais em Uso:*** Atributo responsável por identificar riscos comerciais dentro de um componente, como quebra de patentes (Kalaimagal e Srinivasan (2010)).
- ***Risco em Uso Público:*** Atributo responsável por identificar riscos ao utilizar o componente em lugares públicos (Kalaimagal e Srinivasan (2010)).
- ***Risco do Software:*** Atributo responsável por identificar riscos ao inserir o componente dentro de um projeto de software (Kalaimagal e Srinivasan (2010)).
- ***Risco Operador em Uso:*** Atributo responsável por identificar riscos que o componente pode causar ao sistema quando for utilizado pelo usuário (Kalaimagal e Srinivasan (2010)).
- ***Vulnerabilidade:*** Atributo responsável por indicar se o componente possui mecanismos que o torne seguro a determinados tipos de ataques (Simão e Belchior (2002)).

O Quadro 4-2 apresenta um resumo dos modelos versus atributos, identificando três atributos, *auditabilidade*, *autenticação* e *criptografia de dados* que mais aparecem nos modelos. Um ponto em comum de todos é que eles falham ao não aprofundarem a avaliação desses atributos. Se um arquiteto de software precisar de maiores detalhes sobre segurança para projetar um componente, as descrições realizadas pelos modelos não serão suficientes, consistindo em um ponto muito importante a ser explorado para melhorar a qualidade.

Quadro 4-2. Quadro Comparativo de Atributos entre os Modelos de Qualidade

Atributos de Segurança	ISO/IEC 9126 (2001)	ISO/IEC 25010 (2005)	Bertoa e Vallecillo (2002)	Simão e Belchior (2002)	Rawashdeh e Matalkah (2006)	Andreou e Tziakouris (2007)	Colombo et al. (2007)	Choi et al. (2008)	Sharma, Kumar e Grover(2008)	Alvaro (2009)	Kalaimagal e Srinivasan (2010)
Auditabilidade			X	X		X	X			X	
Autenticidade		X									
Autenticação	X		X			X	X			X	X
Confidencialidade		X		X							
Criptografia de Dados			X			X	X			X	X
Integridade		X									
Não-Repúdio		X									
Privacidade				X							
Resistência a Escalabilidade de Privilégios						X					
Responsabilização		X									
Riscos Comerciais em Uso											X
Risco de Uso Público											X
Risco do Software											X
Risco Operador em Uso											X
Vulnerabilidade				X							

4.2.2 Busca de Características na Área de Segurança

A subetapa anterior consistiu em um bom levantamento sobre os atributos existentes dentro dos modelos de qualidade, identificando áreas de segurança com grande potencial como Auditabilidade, Autenticação e Criptografia de Dados.

Para a busca de artigos e trabalhos dentro da área de segurança foi realizada uma busca sobre as bases de estudo e livros de grandes áreas da segurança da

informação. Além dos atributos identificados na etapa anterior o material encontrado destaca duas outras áreas.

A primeira área é a validação e padronização da entrada de dados dentro do componente, pois a partir da correta mensuração dos dados, se tem a garantia que um componente terá dados corretos para processar, evitando erros. Com esses erros é destacada a segunda grande área de segurança que é a parte de ameaças e vulnerabilidades ao software.

A parte de segurança para hardware e suas configurações não foi incluída neste trabalho por se tratar de algo que não está relacionado com o tema de software.

Após obter esses trabalhos e destacar essas grandes áreas, um professor doutor especialista em segurança da informação analisou os pontos levantados e repassou diversos trabalhos pontuais da área de segurança específico para essas áreas encontradas. Nesse momento iniciou-se a construção da segunda parte do capitulo 2 que retrata os aspectos de segurança dentro dos componentes de software, criando um embasamento teórico grande para a produção do modelo descrito nesse trabalho.

Assim como resultando final dessa etapa foram levantadas as cinco características importantes para o modelo de qualidade deste trabalho são: Criptografia de Dados, Autenticação, Auditabilidade, Controle de Vulnerabilidades e Manipulação de Dados.

4.3 Definir o Modelo de Qualidade para Características Internas de Segurança de Componentes de Software

Essa etapa realiza a explanação sobre o modelo e sua construção. Como se pode observar, esse trabalho não tem o objetivo de gerar um modelo completo de características para componentes de software e sim construir um submodelo da parte de segurança destacando os aspectos importantes relacionados à segurança da informação dentro do contexto de componentes de software.

Essa construção do modelo se deu através da análise do material obtido na etapa anterior sobre as grandes áreas da segurança da informação. A análise desse material ocorreu de forma a buscar os itens mais importantes dentro das características separadas na etapa anterior, e com isso conseguir descrever pontos

necessários para a criação de um modelo que ajudasse a arquitetos e desenvolvedores a buscar uma maior qualidade para seus produtos.

Os próximos itens descreverão em mais detalhes cada característica, subcaracterística e atributo do modelo, destacando os tipos e possíveis formas de avaliação de cada item.

4.3.1 Característica de Criptografia de Dados

Criptografia de dados é o estudo de técnicas matemáticas relacionadas com o aspectos de segurança da informação como confidencialidade, integridade, autenticação de entidades e autenticação de dados de origem (MENEZES ET AL., 1996). O ponto principal dessa característica é a preservação do sigilo dos dados que são enviados/recebidos pelo componente através de meios de comunicação não seguros.

De acordo com Menezes et al. (1996) ao planejar a utilização dessa característica dentro de um software, existem dois itens a serem observados: a escolha do algoritmo criptográfico e a forma de armazenar as chaves do algoritmo.

4.3.1.1 Subcaracterística Algoritmo Criptográfico

A escolha do algoritmo criptográfico é um ponto essencial para a inclusão da característica de criptografia de dados dentro de um componente de software, sendo necessária uma boa análise de quais são as necessidades e gargalos do componente. Para melhorar a forma de gerar essa análise foram elencados alguns pontos importantes para a escolha do melhor algoritmo que está compatibilizado com o planejamento e necessidades dos requisitos do componente. Esses pontos serão descritos nos atributos abaixo.

4.3.1.1.1 Atributo Nome do Algoritmo

Esse atributo tem como simples pretensão possuir o nome do algoritmo ou dos algoritmos utilizados pelo componente, facilitando a busca por maiores informações.

Tipo: String

Valor: Nome do algoritmo criptográfico como AES, 3DES, blowfish, Elgamal, RSA, Diffie-Hellman, entre outros.

4.3.1.1.2 Atributo Tipo Algoritmo

Esse atributo tem a responsabilidade de indicar qual é o tipo do algoritmo de criptografia que será utilizado. Assim esse atributo tem como valores os algoritmos simétricos e os algoritmos assimétricos.

Os algoritmos simétricos possuem a propriedade uma tupla de chaves {*e*, *d*} utilizadas para cifrar/decifrar uma mensagem (GOLLMANN, 2006), porém ambas são secretas. Ao enviar uma mensagem para um destinatário, este deve também receber a chave ***e***, para poder gerar a chave de decifragem ***d***. A vantagem é a obtenção de altas taxas de transferência de dados, sendo o tamanho das chaves curtas quando comparadas com as chaves assimétricas. As desvantagens se encontram na interceptação da chave ***e*** por um atacante e com uma pequena criptoanálise na chave pode identificar o algoritmo utilizado e com ele gerar a chave ***d*** para obter o conteúdo da mensagem. Outra desvantagem está na forma de tratamento com a chave ***e*** pelo destinatário da mensagem, ou seja, se ele irá manter a chave segura para evitar uma interceptação desta também (MENEZES ET AL., 1996).

Os algoritmos assimétricos possuem uma tupla de chaves {*e*, *d*}, a chave ***e*** é chamada de chave pública e a chave ***d*** chamada de chave privada. A vantagem desse algoritmo é que qualquer pessoa que possua a chave ***e*** pode enviar uma mensagem para o dono desta chave, pois somente este possui o domínio da chave ***d***. Outra vantagem é que estas chaves podem passar anos sem que seja necessário mudá-la. Em compensação nenhum algoritmo assimétrico conseguiu provar que é totalmente seguro, podendo ocorrer quebras de código sobre os algoritmos e assim obter o segredo da geração da chave (GOLLMANN, 2006).

A forma de funcionamento de cada algoritmo é descrita com maiores detalhes na seção 2.5.2 no capitulo 2.

Tipo: String

Valor: Tipo do algoritmo criptográfico com o valor Simétrico, Assimétrico.

4.3.1.1.3 Atributo Tamanho da Chave Criptográfica

Esse atributo possui a responsabilidade de mensurar o tamanho da chave criptográfica gerada pelo algoritmo. O tamanho da chave de um algoritmo criptográfico é medido em bits (MENEZES ET AL., 1996), mas esse não significa o

tamanho da segurança que a chave possui, já que existem formas de ataque que podem reduzir esse grau de segurança quebrando a chave ao utilizar menos bits (NIST, 2012), como por exemplo, o algoritmo Triple DES que possui uma chave criptográfica de tamanho de 168 bits, mas possui uma segurança de até 112 bits, já que possui uma forma de ataque de 2^{112} bits conhecida (GOLLMANN, 2006).

Dois aspectos devem ser observados na hora de escolher o melhor tamanho de chave para o componente de software: o tamanho da segurança que essa chave proporciona e qual é o poder de processamento que seu aplicativo irá suportar, já que dispositivos como celulares possuem uma memória limitada.

Para algoritmos simétricos Nist (2012) recomenda que o tamanho de chave mínimo para uma segurança adequada seja a partir de 112 bits, como o algoritmo Triple DES, sendo o mais comum o uso de chave com tamanho de 128 bits como o AES. O governo estadunidense utiliza chaves AES com tamanho de 192 ou 256 bits para documentos mais sensíveis.

Para os algoritmos assimétricos a contagem diferencia um pouco dos algoritmos simétricos. Nist (2012) coloca que chaves RSA de 1024 bits possuem a mesma segurança que chaves de algoritmos simétricos de 80 bits. Já as chaves RSA de 2048 bits possuem a mesma segurança que chaves simétricas de 112 bits e por fim chaves RSA de 3072 bits possuem a mesma segurança que chaves de 128 bits. O recomendado por eles é que sejam utilizados algoritmos com chaves de tamanho de 2048 bits.

Para os algoritmos assimétricos ECC (*Elliptic Curve Cryptography*) possuem uma codificação de tamanho de chaves que se caracteriza por ser o dobro dos algoritmos simétricos, por exemplo, uma chave ECC de tamanho de 224 bits possui o mesmo nível de segurança que uma chave simétrica de 112 bits. O mesmo documento recomenda o uso de chaves ECC com tamanho de 224 bits.

Um site que reúne o tamanho das chaves recomendadas por órgãos privados e órgãos governamentais é o *keylength* (KEYLENGTH, 2012). Este site possui uma lista de recomendações de tamanhos de chaves e tempo de vida de várias formas de algoritmos, boas para serem consultadas por arquitetos e desenvolvedores.

Tipo: Inteiro

Valor: Valor numérico positivo em bits.

4.3.1.1.4 Atributo Vida Útil da Chave Criptográfica

Esse atributo possui a responsabilidade de mensurar o tempo de vida útil de uma chave criptográfica que é chamado de *cryptoperiod* ou período de cifragem. A sua definição consiste no tempo que uma determinada chave é autorizada para o uso, por entidades legitimas (NIST, 2012). Em geral pequenos períodos de cifragem melhoram a segurança, por exemplo, para proteger dados, pois caso seja realizada uma quebra da chave poucos dados serão comprometidos.

Por outro lado, para realizar a distribuição das chaves públicas de um algoritmo assimétrico é interessante ter um período de cifragem maior, para não se tornar incômodo a troca a cada pouco tempo das chaves públicas. O período de cifragem é constituído de dois períodos. O primeiro é o período de tempo de uso pelo dono da chave, ou seja, o período de tempo em que mensagens são protegidas por determinada chave. O segundo é o período de tempo de uso pelo benefício da chave, ou seja, o período de tempo em que alguém que possua a chave pode utiliza-la em mensagens já codificadas pelo primeiro período de tempo, em geral visualizar mensagens antigas (NIST, 2012).

Em média as chaves criptográficas têm uma vida útil de 1 a 2 anos. Um documento bem produzido, que foi utilizado para referencias deste trabalho é o (NIST, 2012). No tópico sobre *cryptoperiod* há uma tabela indicando o tempo de vida de vários modelos de chaves, é uma boa forma de planejar a vida útil das chaves para seu componente de software. E como já citado no atributo anterior um bom site para verificar o tempo de vida de cada chave criptográfica é o *keylength* (KEYLENGTH, 2012).

Tipo: Inteiro + String

Valor: Valor numérico representando uma unidade de tempo; String representa a unidade de tempo, podendo ser milissegundos, segundos, horas, dias, meses e anos.

4.3.1.1.5 Atributo Custo Computacional de Produção da Chave Criptográfica

Esse atributo possui a responsabilidade de mensurar o custo computacional para a produção de uma nova chave criptográfica, seja para revogação ou criação de uma nova identidade. Isso é algo que pode interferir na produtividade de um componente de software. Os algoritmos assimétricos atuais possuem um custo

computacional muito mais caro quando comparada com algoritmos simétricos de segurança equivalente (FERGUSON; SCHNEIER; KOHNO, 2010).

Existem algumas questões relacionadas à eficiência que corroboram essa indicação. O primeiro ponto está na publicação da chave pública assimétrica, pois é necessário publicá-la sem revelar traços da chave privada assimétrica. Para isso é necessária uma matemática muito mais desenvolvida que a de “simples” algoritmos simétricos, que tendem a manter ambas as chaves secretas (MENEZES ET AL., 1996). Assim o custo computacional da chave assimétrica tende a ser maior, deixando a produção da nova chave mais lenta.

A geração das chaves assimétricas RSA e DSA envolvem um pré-processamento muito grande. Para gerar uma chave RSA de 512 bits é necessário que o algoritmo ache dois números primos de 256 bits. Para gerar uma chave DSA de 512 bits é necessário um número primo de 512 bits (NIST, 2012). Em nível de comparação a segunda chave precisa de mais processamento que a primeira chave, pois achar um número primo de 512 bits é muito mais caro que dois números de 256 bits. É necessário levar em consideração também o tamanho da chave, pois quando maior a chave maior a necessidade das operações e recursos para sua produção.

Tipo: Inteiro + String

Valor: Valor numérico representando uma unidade de tempo; String representa a unidade de tempo, podendo ser milissegundos, segundos, horas, dias, meses e anos.

4.3.1.1.6 Atributo Custo Computacional de Operação da Chave Criptográfica

Esse atributo possui a responsabilidade de mensurar o custo computacional para as operações com chave criptográfica. Este custo tende a seguir o mesmo padrão da geração de novas chaves, ou seja, algoritmos assimétricos serão mais lentos para gerar dados criptografados que os algoritmos simétricos.

Isso ocorre novamente pela natureza da chave do algoritmo assimétrico, pois a criptografia de uma das chaves é pública e desta forma visível a olhos atacantes, podendo ser utilizada por atacantes para produzir mensagens arbitrárias e sobre elas realizar criptoanálises. Assim se o algoritmo assimétrico for determinístico, ou seja, possuir um conjunto de dados ou ações repetidas, o atacante pode executar uma pesquisa exaustiva sobre os dados criptografados que ainda possuem uma

estrutura e estão sujeitas a busca, portanto um algoritmo assimétrico deve incluir aleatoriedade extra no código gerado, implicando no aumento do volume de dados (NIST, 2012).

Por exemplo, o algoritmo assimétrico RSA com uma chave de 1024 bits pode cifrar um elemento de dados com até 117 bytes, resultando em um valor de 128 bytes. Caso fosse necessário cifrar um arquivo grande utilizando apenas o algoritmo RSA, provavelmente o tamanho da mensagem criptografada seria 9% maior que a mensagem original, inserindo 450 megabytes a mais em um arquivo de 5 gigabytes (FERGUSON; SCHNEIER; KOHNO, 2010). Os algoritmos simétricos acrescentam um valor de tamanho constante sobre a mensagem, como 32 bytes a mais para os algoritmos AES.

Ambos os tipos de algoritmos trabalham com o conceito de cifradores por bloco, ou seja, é cifrado um bloco de texto de N bits. Um bom cifrador deve criar um novo bloco cifrado que mistura bits do bloco de texto e da chave criptográfica, de modo que quase impossibilita a recuperação do texto cifrado. Para os algoritmos simétricos conseguirem realizar essa mistura de bits utilizam-se algumas operações como o xor, módulos de rotação de 2^{32} ou 2^{64} bits e outras operações simples. O ponto comum dessas operações é que são operações rápidas de serem processadas, chegando a bilhões destas por segundo (RIFÀ-POUS; HERRERA-JOANCOMARTÍ, 2011).

Uma forma de contornar esses problemas de custo computacional se encontra na utilização de sistemas de criptografia híbridos. Nesses sistemas uma chave simétrica (chave de sessão) é gerada por uma parte e esta é então criptografada pela chave pública de cada destinatário e enviada para este. Cada destinatário usa a chave simétrica correspondente para decifrar a chave de sessão. Uma vez que ambas as partes possuam a chave de sessão, utiliza-se o algoritmo simétrico para cifrar/decifrar mensagens.

Tipo: Inteiro + String

Valor: Valor numérico representando uma unidade de tempo; String representa a unidade de tempo, podendo ser milissegundos, segundos, horas, dias, meses e anos.

4.3.1.2 Subcaracterística Armazenamento da Chave Criptográfica

Essa subcaracterística é responsável pelo armazenamento da chave criptográfica, devendo este ser seguro evitando a obtenção ilícita da chave.

4.3.1.2.1 Atributo Forma

Esse atributo possui a responsabilidade de definir uma forma de armazenamento para a chave criptográfica. Para cada forma de chave criptográfica ou característica desejada existem diversas formas de armazená-la. Por exemplo, no caso de chaves assimétricas o armazenamento ocorre através de um arquivo FEK (File Encryption Key) e de um arquivo FSK (File Signature Key) geralmente gerados durante a criação da chave. Alguns exemplos de serviços de armazenamento e distribuição que fazem isso para chaves assimétricas são o SiRiUS, Cepheus, SNAD e o EFS. Já para chaves simétricas alguns exemplos de serviços de armazenamento e distribuição são o Needham-Schroeder, Leighton-Micali e o Blom. Mais detalhes sobre todos esses serviços estão descritos no estudo realizado por (CHEN; ZHOU, 2010).

Tipo: String

Valor: Representa o formato de armazenamento da chave como SiRiUS, Cepheus, SNAD, EFS, Needham-Schroeder, Leighton-Micali e o Blom.

4.3.1.3 Considerações Finais

Para finalizar a característica de Criptografia de Dados será realizada uma descrição sucinta de cada subcaracterística com os atributos e tipos destes. O Quadro 4-3 mostra todos os itens que compõem a característica.

Quadro 4-3. Descrição da Característica Criptografia de Dados

Característica	Subcaracterística	Atributo	Tipo	Valor
Criptografia de Dados	Algoritmo	Nome	String	AES, 3DES, blowfish, Elgamal, RSA, Diffie-Hellman
		Tipo Algoritmo	String	simétricos e assimétricos
		Tamanho da Chave	Inteiro	positivo em bits
		Vida Útil	Inteiro/ String	tempo/unidade de tempo
		Custo Computacional Produção	Inteiro/ String	tempo/unidade de tempo
		Custo Computacional Operação	Inteiro/ String	tempo/unidade de tempo
	Armazenamento Chave	Forma	String	SiRiUS, Cepheus, SNAD, EFS

Para a característica de Criptografia de Dados foi definido duas subcaracterísticas: *Algoritmo* que representa qual foi o algoritmo criptográfico escolhido pelo usuário, onde este pode ser mais do que um algoritmo. *Armazenamento Chave* que representa a forma de armazenamento da chave criptográfica. Dentro da subcaracterística Algoritmo foram definidos cinco atributos com tipos e valores:

- **Nome**: Atributo que define o nome do algoritmo criptográfico utilizado. É representado por um atributo do tipo String, contendo um ou mais algoritmos. O seu valor são os algoritmos simétricos e assimétricos, como por exemplo, o AES, 3DES, blowfish, Elgamal, RSA, Diffie-Hellman.
- **Tipo Algoritmo**: Atributo que define o tipo do algoritmo criptográfico. É representado por um atributo do tipo String. O seu valor é Simétrico ou Assimétrico.
- **Tamanho da Chave**: Atributo que define qual é o tamanho da chave criptográfica. É representado por um atributo do tipo Inteiro. O seu valor é um número positivo em bits.
- **Vida Útil**: Atributo que define qual é a vida útil de uma chave criptográfica. É representada por um atributo composto de Inteiro e String, onde o primeiro representa um valor de tempo e a String a unidade de medida do tempo.
- **Custo Computacional Produção**: Atributo que define qual é o custo necessário para gerar uma chave criptográfica, como o clock de processamento é variável é mais fácil representar pelo tempo. É representada por um atributo composto de Inteiro e String, onde o primeiro representa um valor de tempo e a String a unidade de medida do tempo.
- **Custo Computacional Operação**: Atributo que define qual é o custo necessário para realizar uma operação com a chave, como o clock de processamento pode variar é mais fácil representar pelo tempo. É representada por um atributo composto de Inteiro e String, onde o primeiro representa um valor de tempo e a String a unidade de medida do tempo.

Dentro da subcaracterística *Armazenamento Chave* foi definido apenas um atributo com seu tipo e valor:

- **Forma**: Atributo que define a forma de armazenamento da chave criptográfica. É representado por um atributo do tipo String. O seu valor pode estar relacionas com os serviços SiRiUS, Cepheus, SNAD, EFS, Needham-Schroeder, Leighton-Micali e o Blom.

4.3.2 Característica de Autenticação

Autenticação é a identificação ou autenticação de entidade em um processo onde uma parte é assegurada, via aquisição de alguma evidência corroborativa, da identidade de uma segunda parte envolvida no protocolo, e que a segunda parte está atualmente participando, ou seja, é ativa no momento em que a evidência foi adquirida (MENEZES ET AL., 1996). Essa característica contempla a identificação do usuário e a sua correta autenticação.

Essa característica permite ao desenvolvedor criar interfaces de entrada para o seu componente. Essas interfaces de acesso devem realizar determinadas funções para verificar se o usuário que deseja entrar no sistema possui o acesso. Para tanto existem diversas formas de garantir a veracidade de uma identidade a partir de classes de identificação. Outro ponto é a assinatura digital de um documento verificando se aquele usuário é realmente dono ou o produziu.

4.3.2.1 Subcaracterística Classe de Identificação

A subcaracterística Classe de Identificação descreve as formas que existem para um usuário realizar uma identificação em um sistema. Essas classes de identificação estão relacionadas a três pontos principais, algo que o usuário conhece ou algo que o usuário possui ou algo que o usuário tem inerente.

4.3.2.1.1 Atributo Algo Conhecido

Esse atributo possui a responsabilidade de definir uma classe a partir do conhecimento do usuário. A definição básica desse atributo é algo que o individuo conhece e que pode ser utilizado para realizar o ingresso em um sistema (MENEZES ET AL., 1996). Esse “algo” consiste em um valor armazenado pelo usuário, sendo utilizado para a identificação no sistema. Algumas formas de representar essa classe são senhas, passphrases, senhas únicas e identificações por desafio resposta. Normalmente as senhas e senhas únicas são consideradas de

baixa segurança, já os passphrases e identificações por desafio resposta são considerados de segurança intermediária (MENEZES ET AL., 1996). Para maiores detalhes verifique a seção 2.5.3.

Tipo: String

Valor: Valor da forma de representação da classe como senhas, passphrases.

4.3.2.1.2 Atributo Algo de Posse

Esse atributo possui a responsabilidade de definir uma classe a partir da posse de algum item pelo usuário. A definição básica desse atributo é algo que o individuo possui normalmente um acessório físico com função semelhante à de um passaporte (MENEZES ET AL., 1996). Esse acessório físico normalmente é criado por um determinado setor especializado da empresa ou por terceiros que realizam esse trabalho como certificadores digitais. Algumas formas de representar essa classe são os cartões magnéticos, cartões com chip e tokens, os quais são considerados de segurança intermediária. Para maiores detalhes verifique a seção 2.5.3.

Tipo: String

Valor: Valor da forma de representação da classe como cartões magnéticos e tokens.

4.3.2.1.3 Atributo Algo Inerente

Esse atributo possui a responsabilidade de definir uma classe a partir de características inerentes do individuo. A definição básica desse atributo é algo que o individuo possui de intrínseco fazendo com que seja único. Esses itens podem ser características físicas como impressão digital, Iris, voz ou características involuntárias como assinaturas, forma de escrita (WINDLEY, 2005). Esse atributo possui o problema do falso-positivo e falso-negativo, que é quando um usuário consegue se identificar mas não tem cadastro ou o contrário. A maioria das formas dessa categoria possui uma segurança considerada de intermediária para alta.

É interessante notar que essa classe não é mantida sempre em segredo como a classe de conhecimento, mas está sempre com a pessoa, como a classe de posse. É interessante notar que essa classe é muito parecida com a carteira de

motorista, o número é publico, mas a dificuldade em falsificá-la é grande. Para maiores detalhes verifique a seção 2.5.3.

Tipo: String

Valor: Valor da forma de representação da classe como impressões digitais, iris, voz, assinaturas, formas de escrita.

4.3.2.2 Subcaracterística Combinação de Classes

A subcaracterística Combinação de Classes consiste na combinação das formas que existem para realizar a identificação de um usuário em um sistema qualquer. Essas classes de identificação estão relacionadas a dois atributos, a lista de classes com todos os itens necessários na visão do usuário e a robustez dessa combinação.

4.3.2.2.1 Atributo Lista de Formas

Esse atributo possui a responsabilidade de enumerar a combinação das subcaracterísticas de classes de identificação, onde o usuário escolhe duas ou mais formas de identificação para deixar mais forte a segurança da identificação do usuário. Um ponto a ser observado é que a combinação de duas formas pertencentes a mesma classe não produz um resultando muito bom se comparado com formas de classes diferentes.

Tipo: String

Valor: Valor da forma de representação das classes de identificação como senhas, cartões magnéticos e impressões digitais.

4.3.2.2.2 Atributo Robustez

Esse atributo possui a responsabilidade de criar um nível de robustez das combinações realizadas no atributo anterior. O' Gorman (2003) criou uma relação de destaque entre as combinações o Quadro 4-4 realiza esse destaque de combinações.

Dependendo das formas combinadas estas podem gerar uma segurança mais alta, assim podem ser tratadas como seguranças baixas, intermediárias, altas e muito altas.

Quadro 4-4. Combinação de classes de identificação (O'GORMAN, 2003)

Combinação	Vantagens	Desvantagens	Exemplo
Classe Conhecido + Classe Posse	Perda/Roubo do Token protegido por senha	Necessário carregar o token e memorizar senha	Cartão com chip
Classe Posse + Classe Inerente	Perda/Roubo do Token protegido pelo ID	Necessário carregar o token, mas não o ID se este for biométrico	Identidade
Classe Conhecido + Classe Inerente	Dois fatores provem a segurança em caso de comprometimento	Necessário memorizar a senha e ter um ID	Acesso biométrico com senha
Classe Conhecido + Classe Posse + Classe Inerente	Um terceiro fator para prover segurança caso os outros fatores forem comprometidos	Necessário memorizar senha, ter um ID e carregar o token	Aplicações militares querem uma Identidade, checada por um guarda mais uma senha

Tipo: String

Valor: Valores representando o nível de segurança de cada combinação variam entre o baixo, intermediário, alto e muito alto.

4.3.2.3 Subcaracterística Algoritmos de Funções Hash

A subcaracterística Algoritmos de Funções Hash são uma importante parte da criptografia com objetivo específico de integridade de dados e autenticação de mensagens (MENEZES ET AL., 1996). Além disso, ela serve para gerar Assinaturas Digitais. A geração dessas assinaturas é descrita por Windley (2005): É gerado um *código-hash* do documento a ser assinado digitalmente, esse código é cifrado com uma chave privada de um algoritmo assimétrico e enviado com o documento não criptografado. Chegando ao destino o número é decifrado com a chave pública e é comparado com o resultado da função *hash* sobre o documento. Se forem iguais o documento não foi alterado. Para avaliar os pontos de um algoritmo Hash corretamente este foi dividido em quatro atributos que serão descritos abaixo.

4.3.2.3.1 Atributo Nome

Esse atributo tem como simples pretensão possuir o nome do algoritmo ou dos algoritmos utilizados pelo componente, facilitando a busca por maiores informações.

Tipo: String

Valor: Nome do algoritmo de hash como o MD5, RIPEMD-160, SHA1, SHA256, SHA384, SHA512.

4.3.2.3.2 Atributo Tamanho Código-Hash

Esse atributo possui a responsabilidade de mensurar o tamanho da saída do código-hash gerado pelo algoritmo, evitando colisões de códigos (CALLAS, 2006). Esse tamanho é medido em bits, mas da mesma forma que as chaves criptográficas quando maior o tamanho do código-hash, maior o nivel de processamento necessário para criar o código, consumindo mais recursos do hardware e tempo de processamento.

Alguns exemplos de tamanho de código-hash são: MD5, com geração de código de 128 bits; O SHA1 e o RIPEMD-160 com geração de *código-hash* de 160 bits; SHA-256, SHA 384 e SHA512 com o código já descrito no nome.

Tipo: Inteiro

Valor: Valor numérico positivo em bits.

4.3.2.3.3 Atributo Tamanho do Bloco

Assim como os algoritmos criptográficos, os algoritmos hash quebram uma mensagem em pequenos trechos chamados blocos. Esse atributo possui a responsabilidade de mensurar o tamanho dos blocos de informação utilizados para gerar o código hash.

Esses blocos são medidos em bits e cada algoritmo define o tamanho dos blocos. Existem algoritmos que usam blocos de 128 bits como o MD2; blocos de 256 bits como GOST e o PANAMA; Mas atualmente a grande maioria utiliza blocos de 512 bits como o SHA1, RIPEMD-160, MD5 e ainda existem algoritmos que utilizam blocos de 1 kbit como o SHA512 (NIST, 2012).

Tipo: Inteiro

Valor: Valor numérico positivo em bits.

4.3.2.3.4 Atributo Custo Computacional produção

Esse atributo possui a responsabilidade de mensurar o custo da produção de um código-hash. A construção de um código-hash consiste em dois elementos, uma função de compressão que a partir de uma entrada de um tamanho fixo gera uma

saída e um extensor de domínio que obtêm as saídas da função de compressão e com uma estrutura iterativa gera o código-hash. De acordo com Rifà-Pous e Herrera-Joancomartí (2011), o custo de inicialização do algoritmo é pequeno assim quanto maior o bloco de entrada para a função hash e o tamanho do código-hash em bits, menor poder de processamento possui um dispositivo. Dessa forma o tamanho do código e o tempo de processamento são diretamente proporcionais.

Tipo: Inteiro + String

Valor: Valor numérico representando uma unidade de tempo; String representa a unidade de tempo, podendo ser milissegundos, segundos, horas, dias, meses e anos.

4.3.2.4 Considerações Finais

Para finalizar a característica de Autenticação será realizada uma descrição sucinta de cada subcaracterística com os atributos e tipos destes. O Quadro 4-5 mostra todos os itens que compõem a característica.

Para a característica de Autenticação foi definido três subcaracterísticas: *Classe de Identificação* que representa as diferentes classes existentes para a identificação de um usuário. *Combinação de Classes* que representa a combinação entre classes de identificação e *Algoritmos de Funções Hash Criptográficas* que representam as funções hash que verificam assinaturas digitais.

Quadro 4-5. Descrição da Característica Autenticação

Característica	Subcaracterística	Atributo	Tipo	Valor
Autenticação	Classe de Identificação	Algo Conhecido	String	senhas, PINs, desafios-respostas
		Algo de Posse	String	cartões magnéticos, cartões com chip, tokens
		Algo Inerente	String	biometria, forma de escrita
	Combinação de Classes	Lista de Formas	String	combinação de itens das três classes
		Robustez	String	baixo, intermediário, alto, muito alto
	Algoritmos de Funções Hash	Nome	String	MD5, SHA1, SHA2
		Tamanho Código-Hash	Inteiro	positivo em bits
		Tamanho do Bloco	Inteiro	positivo em bits
		Custo Computacional Produção	Inteiro/ String	tempo/unidade de Tempo

A subcaracterística de Classe de Identificação possui três atributos com tipos e valores:

- **Algo Conhecido**: Atributo que define a classe de identificação que se baseia no conhecimento de algo pelo usuário para a identificação. É representado por um atributo do tipo String. O seu valor são a senha, passphrases, senhas únicas e identificações por desafio resposta.
- **Algo de Posse**: Atributo que define a classe de identificação que se baseia em algo que o usuário possui como um item físico. É representado por um atributo do tipo String. O seu valor são cartões magnéticos, cartões com chip e tokens.
- **Algo Inerente**: Atributo que define a classe de identificação que se baseia em algo intrínseco ao usuário, fazendo com que seja único. É representado por um atributo do tipo String. O seu valor são impressões digitais, iris, assinaturas, formas de escrita.

Dentro da subcaracterística *Combinação de Classes* foram definidos dois atributos com seu tipo e valor:

- **Lista de Formas**: Atributo que define uma lista de formas das três classes de identificação. É representado por um atributo do tipo String, contendo duas ou mais formas. O seu valor são as formas como senha, cartões magnéticos e impressão digital.
- **Robustez**: Representa o nível de segurança resultante da combinação de duas ou mais formas das classes de identificação. É representado por um atributo do tipo String. O seu valor está nos níveis, baixo, intermediário, alto e muito alto.

Dentro da subcaracterística *Algoritmos de Funções Hash* foram definidos quatro atributos com seu tipo e valor:

- **Nome**: Atributo que define o nome do algoritmo criptográfico utilizado. É representado por um atributo do tipo String, contendo um ou mais algoritmos. O seu valor são os algoritmos MD5, RIPEMD-160, SHA1, SHA256, SHA384, SHA512.
- **Tamanho Código-Hash**: Atributo que define o tamanho máximo da geração de um código-hash. Quando maior o tamanho menor é o caso de

colisões, mas é necessário um maior processamento. É representado por um atributo do tipo Inteiro. O seu valor é um número positivo em bits.

- **Tamanho do Bloco**: Atributo que define o tamanho do bloco de informações que o algoritmo usa para gerar o código-hash. É representado por um atributo do tipo Inteiro. O seu valor é um número positivo em bits.
- **Custo Computacional Produção**: Atributo que define qual é o custo necessário para gerar um código-hash, como o clock de processamento é variável é mais fácil representar pelo tempo. É representada por um atributo composto de Inteiro e String, onde o primeiro representa um valor de tempo e a String a unidade de medida do tempo.

4.3.3 Característica de Auditabilidade

Conforme definido a Auditabilidade é o processo de preservação das qualidades de Não repúdio e Responsabilidade através de um mecanismo de auditoria, com capacidade de gravar acessos ao sistema e seus dados. Por padrão, o componente pode prover funcionalidades para gravar cada operação realizada por seus usuários com os dados acessados (BERTOA; VALLECILLO, 2002). Assim a auditabilidade é necessária para um controle sobre as ações dos usuários sobre o componente e verificação de erros no sistema.

Ao planejar a utilização dessa característica dentro de um componente de software, existem quatro itens a serem observados: o formato do log, a forma de armazenamento dos arquivos de log, as operações de arquivamento e as operações de análise.

4.3.3.1 Subcaracterística Formato Log

A escolha do formato do arquivo de log é um ponto importante para evitar que seja formada uma torre de Babel nos sistemas de uma empresa. Pois tendo um mesmo padrão ou padrões de arquivo de log, é possível aperfeiçoar operações de auditabilidade.

4.3.3.1.1 Atributo Classe de Eventos

Esse atributo possui a responsabilidade de mensurar o grau de importância que determinado evento causou no sistema. O padrão syslog (IETF, 2009) possui oito graus para escalar um evento. Emergency, Alert, Critical, Error, Warning, Notifce, Info, Debug. Assim para cada evento que ocorrer no sistema esse grau ajuda a diagnosticar problemas mais facilmente.

Tipo: String

Valor: Define o grau de intensidade do evento, como Emergency, Alert, Critical, Error, Warning, Notifce, Info, Debug.

4.3.3.1.2 Atributo Formato

Esse atributo possui a responsabilidade de definir o padrão de formato adotado por um arquivo de log. Um desses formatos é o syslog (IETF, 2009), que possui uma gama de flexibilidades ajudando a criar um evento em um log com mais informações como data, hora, sistema, ip máquina.

Tipo: String

Valor: Define o formato a ser adotado no arquivo de log do componente, como o Syslog.

4.3.3.2 Subcaracterística Forma de Armazenamento

Essa subcaracterística é responsável pelo armazenamento do arquivo de log. Esse armazenamento deve ser seguro evitando a obtenção ilícita do arquivo.

4.3.3.2.1 Atributo Tipo de Armazenamento

Esse atributo possui a responsabilidade de definir a forma de armazenamento de um arquivo de log, já que este possui muitas informações pessoais e dados sigilosos de usuários. Sans (2002) descreve algumas formas de realizar esse arquivamento como a impressão dos arquivos de log, armazenamento em dispositivos WORM (*Write Once, Read Many*), armazenamento em partições separadas, utilização de linhas seriais com outras máquinas, armazenando os arquivos nessa máquina remota.

Tipo: String

Valor: Representa o formato de armazenamento do arquivo como impressão, dispositivos WORM, partições separadas.

4.3.3.3 Subcaracterística Operação de Armazenamento

Essa subcaracterística é responsável por informar se existe certas operações para o armazenamento do arquivo de log. Além da forma de armazenamento ser segura, estas operações melhoram o processo de armazenamento.

4.3.3.3.1 Atributo Algoritmo Hash

Esse atributo possui a responsabilidade de definir se o mecanismo de log do componente utiliza a funcionalidade de gerar um código-hash para o arquivo de log. Essa operação consiste na criação de um valor *hash* para cada arquivo de log armazenado. Esse valor é armazenado em um ambiente seguro para a verificação de mudanças, detectando assim possíveis manipulações do arquivo (KENT; SOUPPAYA, 2006).

Tipo: String

Valor: Valor que representa o nome do algoritmo hash associado com a função como MD5, SHA1, SHA2.

4.3.3.3.2 Atributo Arquivamento de Arquivo de Log

Esse atributo possui a responsabilidade de definir se o mecanismo de log do componente utiliza a operação de arquivamento de arquivo de log. Essa operação consiste na conservação de um arquivo de log, realizado de acordo com a importância do software e as normas internas. Existem duas formas de realizar a operação. A primeira é a Retenção do Arquivo de Log onde o arquivo é tratado de forma idêntica aos arquivos de log do sistema operacional. A segunda é a Preservação do Arquivo de Log, realizando a manutenção dos arquivos que seriam descartados, por possuírem informações de interesse para investigações (KENT; SOUPPAYA, 2006).

Tipo: Booleano

Valor: Valor que representa se essa operação é realizada no componente

4.3.3.3.3 Atributo Rotação de Arquivo de Log

Esse atributo possui a responsabilidade de definir se o mecanismo de log do componente utiliza a funcionalidade de rotação de arquivo de log. Essa operação

consiste no encerramento de um arquivo de log quando é considerado completo, criando um novo arquivo para armazenar os eventos. Pode ser realizada através de um agendamento (diário, semanal, mensal) ou quando o arquivo atingir um tamanho fixo (KENT; SOUPPAYA, 2006).

Tipo: Booleano

Valor: Valor que representa se essa operação é realizada no componente

4.3.3.3.4 Atributo Compressão de Arquivo de Log

Esse atributo possui a responsabilidade de definir se o mecanismo de log do componente utiliza a funcionalidade de compressão de arquivo de log. Essa operação consiste na compressão de arquivos de log antigos evitando a necessidade de mais espaço de armazenamento. Pode ser realizada durante as operações de Rotação de Arquivo de Log ou Arquivamento de Arquivo de Log (KENT; SOUPPAYA, 2006).

Tipo: Booleano

Valor: Valor que representa se essa operação é realizada no componente

4.3.3.3.5 Atributo Redução de Arquivo de Log

Esse atributo possui a responsabilidade de definir se o mecanismo de log do componente utiliza a funcionalidade de redução de arquivo de log. Essa operação consiste na remoção de entradas desnecessárias dentro de um arquivo de log, gerando um novo arquivo menor. Pode ser realizada antes da operação de Compressão de Arquivo de Log ou da operação de Análise de Arquivo de Log (KENT; SOUPPAYA, 2006).

Tipo: Booleano

Valor: Valor que representa se essa operação é realizada no componente

4.3.3.3.6 Atributo Visualização de Arquivo de Log

Esse atributo possui a responsabilidade de definir se o mecanismo de log do componente utiliza a funcionalidade de visualização de arquivo de log. Essa operação consiste na apresentação de eventos dentro de um arquivo de log em uma forma fácil de ser entendida, podendo prover habilidades para ordenação de entradas (KENT; SOUPPAYA, 2006).

Tipo: Booleano

Valor: Valor que representa se essa operação é realizada no componente

4.3.3.4 Subcaracterística Operação de Análise

Essa subcaracterística é responsável pelas operações de análise de arquivo de log. Essa análise é muito importante para identificar falhas e erros no sistema e realizar uma auditoria sobre a utilização do usuário.

4.3.3.4.1 Atributo Filtragem de Eventos

Esse atributo possui a responsabilidade de definir se o mecanismo de log do componente utiliza a funcionalidade de filtragem de eventos. Essa operação consiste na remoção de eventos contidos em um arquivo de log que não sejam úteis para análise (KENT; SOUPPAYA, 2006).

Tipo: Booleano

Valor: Valor que representa se essa operação é realizada no componente

4.3.3.4.2 Atributo Agregação de Eventos

Esse atributo possui a responsabilidade de definir se o mecanismo de log do componente utiliza a funcionalidade de agregação de eventos. Essa operação consiste no agrupamento em uma única entrada de eventos similares, acrescida do número de quantidade de ocorrências (KENT; SOUPPAYA, 2006).

Tipo: Booleano

Valor: Valor que representa se essa operação é realizada no componente

4.3.3.4.3 Atributo Correlação de Eventos

Esse atributo possui a responsabilidade de definir se o mecanismo de log do componente utiliza a funcionalidade de correlação de eventos. Essa operação consiste na procura por relações entre duas ou mais entradas de um arquivo de log (KENT; SOUPPAYA, 2006).

Tipo: Booleano

Valor: Valor que representa se essa operação é realizada no componente

4.3.3.4.4 Atributo Conversão de Arquivo de Log

Esse atributo possui a responsabilidade de definir se o mecanismo de log do componente utiliza a funcionalidade de conversão de arquivo de log. Essa operação consiste na transformação do formato de um arquivo de log para outro formato para fins de portabilidade (KENT; SOUPPAYA, 2006).

Tipo: Booleano

Valor: Valor que representa se essa operação é realizada no componente

4.3.3.4.5 Atributo Normalização de Arquivo de Log

Esse atributo possui a responsabilidade de definir se o mecanismo de log do componente utiliza a funcionalidade de normalização de arquivo de log. Essa operação consiste na categorização dos eventos de um arquivo de log para uma representação particular (KENT; SOUPPAYA, 2006).

Tipo: Booleano

Valor: Valor que representa se essa operação é realizada no componente

4.3.3.4.6 Atributo Relatório de Arquivo de Log

Esse atributo possui a responsabilidade de definir se o mecanismo de log do componente utiliza a funcionalidade de relatório de arquivo de log. Essa operação consiste na apresentação das análises realizadas, construindo resumos das atividades significativas de um sistema sobre um período de tempo ou verificando informações relacionadas a um evento (KENT; SOUPPAYA, 2006).

Tipo: Booleano

Valor: Valor que representa se essa operação é realizada no componente

4.3.3.5 Considerações Finais

Para finalizar a característica de Auditabilidade será realizada uma descrição sucinta de cada subcaracterística com os atributos e tipos destes. O Quadro 4-6 mostra todos os itens que compõem a característica.

Para a característica de Auditabilidade foram definidas quatro subcaracterísticas: *Formato Log*, que define o padrão a ser utilizado pelo componente para realizar o armazenamento de eventos. *Forma de Armazenamento*, que define a forma de armazenamento dos arquivos de log. *Operação de*

armazenamento que define as operações que podem ser realizadas durante o armazenamento do arquivo de log. *Operação de Análise* que define as operações que podem ser realizadas no processo de análise de um arquivo de log.

Quadro 4-6. Descrição da Característica Auditabilidade

Característica	Subcaracterística	Atributo	Tipo	Valor
Auditabilidade	Formato Log	Classe de Eventos	String	emergency, alert, information, debug
		Formato	String	Syslog
	Armazenamento	Forma	String	impresso, dispositivo WORM, storage
	Operação de Armazenamento	Algoritmo Hash	String	MD5, SHA1, SHA2
		Arquivamento de Arquivo de Log	Booleano	verdadeiro, falso
		Rotação de Arquivo de Log	Booleano	verdadeiro, falso
		Compressão de Arquivo de Log	Booleano	verdadeiro, falso
		Redução de Arquivo de Log	Booleano	verdadeiro, falso
		Visualização de Arquivo de Log	Booleano	verdadeiro, falso
	Operação de Análise	Filtragem de Eventos	Booleano	verdadeiro, falso
		Agregação de Eventos	Booleano	verdadeiro, falso
		Correlação de Eventos	Booleano	verdadeiro, falso
		Conversão de Arquivo de Log	Booleano	verdadeiro, falso
		Normalização de Arquivo de Log	Booleano	verdadeiro, falso
		Relatório de Arquivo de Log	Booleano	verdadeiro, falso

Dentro da subcaracterística Formato Log foram definidos dois atributos com tipos e valores:

- **Classe de Eventos**: Atributo que define o grau de importância que um evento causou ao sistema. É representado por um atributo do tipo String. O seu valor é definido como Emergency, Alert, Critical, Error, Warning, Notifce, Info, Debug.
- **Formato**: Atributo que define o padrão de formato adotado por um arquivo de log. É representado por um atributo do tipo String. O seu valor é um formato padrão de arquivos de log como o syslog.

Dentro da subcaracterística *Armazenamento Chave* foi definido apenas um atributo com seu tipo e valor:

- **Tipo de Armazenamento**: Atributo que define a forma de armazenamento da chave criptográfica. É representado por um atributo do tipo String. O seu valor pode estar relacionas com os serviços SiRiUS, Cepheus, SNAD, EFS, Needham-Schroeder, Leighton-Micali e o Blom.

Dentro da subcaracterística Operação de Armazenamento foram definidos seis atributos com tipos e valores:

- **Algoritmo Hash**: Atributo que define qual é o algoritmo hash utilizado para a operação de verificação de integridade de um arquivo de log. É representado por um atributo do tipo String. O seu valor MD5, SHA1, SHA2.
- **Arquivamento de Arquivo de Log**: Atributo que define se a operação de Arquivamento de Arquivo de Log é realizada pelo mecanismo de log do componente. É representado por um atributo do tipo Booleano. O seu valor é verdadeiro ou falso.
- **Rotação de Arquivo de Log**: Atributo que define se a operação de Rotação de Arquivo de Log é realizada pelo mecanismo de log do componente. É representado por um atributo do tipo Booleano. O seu valor é verdadeiro ou falso.
- **Compressão de Arquivo de Log**: Atributo que define se a operação de Compressão de Arquivo de Log é realizada pelo mecanismo de log do componente. É representado por um atributo do tipo Booleano. O seu valor é verdadeiro ou falso.
- **Redução de Arquivo de Log**: Atributo que define se a operação de Redução de Arquivo de Log é realizada pelo mecanismo de log do componente. É representado por um atributo do tipo Booleano. O seu valor é verdadeiro ou falso.
- **Visualização de Arquivo de Log**: Atributo que define se a operação de Visualização de Arquivo de Log é realizada pelo mecanismo de log do componente. É representado por um atributo do tipo Booleano. O seu valor é verdadeiro ou falso.

Dentro da subcaracterística Operação de Análise foram definidos seis atributos com tipos e valores:

- **Filtragem de Eventos**: Atributo que define se a operação de Filtragem de Eventos é realizada pelo mecanismo de log do componente. É

representado por um atributo do tipo Booleano. O seu valor é verdadeiro ou falso.

- **Agregação de Eventos**: Atributo que define se a operação de Agregação de Eventos é realizada pelo mecanismo de log do componente. É representado por um atributo do tipo Booleano. O seu valor é verdadeiro ou falso.
- **Correlação de Eventos**: Atributo que define se a operação de Correlação de Eventos é realizada pelo mecanismo de log do componente. É representado por um atributo do tipo Booleano. O seu valor é verdadeiro ou falso.
- **Conversão de Arquivo de Log**: Atributo que define se a operação de Conversão de Arquivo de Log é realizada pelo mecanismo de log do componente. É representado por um atributo do tipo Booleano. O seu valor é verdadeiro ou falso.
- **Normalização de Arquivo de Log**: Atributo que define se a operação de Normalização de Arquivo de Log é realizada pelo mecanismo de log do componente. É representado por um atributo do tipo Booleano. O seu valor é verdadeiro ou falso.
- **Relatório de Arquivo de Log**: Atributo que define se a operação de Relatório de Arquivo de Log é realizada pelo mecanismo de log do componente. É representado por um atributo do tipo Booleano. O seu valor é verdadeiro ou falso.

4.3.4 Característica de Controle de Vulnerabilidades

Conforme definido as vulnerabilidades representam um risco para um componente, pois não preserva o correto funcionamento, verificando instâncias de faltas na especificação, desenvolvimento ou configuração deste, tal que a execução de uma dessas instâncias pode violar uma política de segurança da empresa (SHIN; WILLIAMS, 2008). Dessa forma ela é um problema a ser evitado dentro de um componente para que este tenha um bom funcionamento sem falhas que impeçam a execução de suas funcionalidades.

Ao planejar a utilização desta característica dentro de um componente de software existem três pontos importantes a serem observados, a classificação das

vulnerabilidades, o tratamento das vulnerabilidades e o monitoramento das vulnerabilidades.

4.3.4.1 Classificação

A subcaracterística Classificação descreve as informações necessárias para a classificação de uma vulnerabilidade. Essas informações são necessárias para realizar uma classificação dentro de um processo de controle de vulnerabilidades, já que sabendo quais as formas que estas podem atacar, fica mais fácil de armazenar essas informações.

4.3.4.1.1 Atributo Tipo

Esse atributo possui a responsabilidade de definir qual é a classe em que uma vulnerabilidade é classificada. De acordo com Avizienis et al. (2004) existem três tipos de vulnerabilidades, a Falha, que é quando a funcionalidade sofre um desvio de seu funcionamento correto, o Erro, que é a causa de uma falha e a Falta que é a declaração de um erro.

Tipo: String

Valor: Valor que representa o tipo da vulnerabilidade sendo os valores Falha, Erro e Falta.

4.3.4.1.2 Atributo Origem de Faltas

Esse atributo possui a responsabilidade de definir a origem de uma falta. De acordo com Avizienis et al. (2004), existem oito classes que representam pontos de vista para a geração de faltas. Essas fases são as seguintes: Fase de Criação, Fronteiras do Sistema, Causa Fenomenológica, Dimensão, Objetivo, Intenção, Capacidade e Persistência. Conforme o Quadro 6-4 existem subclasses para cada uma das classes citadas acima. Como essas subclasses não são exclusivas entre si, elas podem ser combinadas, gerando assim vários contextos de origem. Mas alguns desses contextos não seriam interessantes, por serem antagônicas como, por exemplo, uma falta maliciosa não pode ser combinada como uma falta não deliberada, pois para causar dano a um sistema esta foi planejada anteriormente.

Tipo: String

Valor: Valor que representa a origem da falta como Fase de Criação, Fronteiras do Sistema, Causa Fenomenológica.

4.3.4.1.3 Atributo Domínio de Falhas

Esse atributo possui a responsabilidade de definir o domínio de falhas. Avizienis et al. (2004) descreve que o domínio de falhas pode ser caracterizado de três formas: Com relação a falhas de conteúdo, com relação a falhas de tempo e com relação a combinação de falhas de conteúdo e falhas de tempo. A Figura 6-4 descreve melhor essas combinações, descrevendo que situações podem ocorrer para cada tipo de falha.

Tipo: String

Valor: Valor que representa o domínio de falhas como falhas com relação ao conteúdo, falhas com relação ao tempo e falhas combinadas de conteúdo e tempo.

4.3.4.2 Tratamento

A subcaracterística de Tratamento descreve as informações necessárias para realizar o tratamento de uma vulnerabilidade, quando esta ocorre. É muito importante um componente possuir uma forma de tratamento de vulnerabilidades para não ficar exposto a ataques e outras formas de obter informações ilícitas pelo componente. Uma técnica é a de detecção, que é a busca por erros e faltas. Outra técnica é a recuperação, que é a transformação do estado de um sistema que contém um ou mais erros e possíveis faltas em um estado sem erros detectados e sem faltas que possam ser ativadas, voltando o sistema a um estado amigável para a utilização do usuário.

4.3.4.2.1 Atributo Detecção

Esse atributo possui a responsabilidade de definir qual é a forma de realizar a detecção das vulnerabilidades. Essa detecção pode ocorrer de duas maneiras de acordo com Avizienis et al. (2004): a detecção de vulnerabilidades latentes também chamadas de faltas dormentes, que são detectadas durante a execução do software pelo cliente. A segunda maneira ocorre quando a execução do software pelo cliente é interrompida para a localização de vulnerabilidades dentro do software, essa segunda maneira é chamada de detecção preemptiva.

Tipo: String

Valor: Valor que representa a forma de realizar a detecção de vulnerabilidades sendo os valores detecção de erros latentes ou detecção preemptiva.

4.3.4.2.2 Atributo Manipulação de Erros

Esse atributo possui a responsabilidade de definir a forma de manipular erros, ou seja, a eliminação de erros presentes dentro do estado de falhas do sistema. Para esse objetivo existem três técnicas principais (AVIZIENIS ET AL., 2004). O *Rollback* que consiste na volta do sistema a um estado salvo anteriormente, chamado de *chechpoint*. O *Rollforward* que consiste na criação de um novo estado para o sistema sem erros. E o *Compensation* que consiste no mascaramento do erro, não o mostrando para o usuário, isso somente pode ocorrer quando o estado errôneo possui redundância suficiente para mascarar o erro. As técnicas de *Rollback* e *Rollforward* podem ser utilizadas em conjunto, nessa ordem, auxiliando ainda mais a manipulação de erros.

Tipo: String

Valor: Valor que representa a forma de manipular erros sendo os valores as técnicas de Rollback, Rollforward e Compensation.

4.3.4.2.3 Atributo Manipulação de Faltas

Esse atributo possui a responsabilidade de definir a forma de manipular faltas, ou seja, a eliminação de faltas presentes dentro do estado de falhas do sistema. Para esse objetivo existem quatro técnicas principais (AVIZIENIS ET AL., 2004). O Diagnóstico que consiste em localizar e identificar itens que estão causando as faltas. O Isolamento que consiste em excluir lógica ou fisicamente os itens com faltas. A Reconfiguração que consiste na troca de itens defeituosos por outros sem faltas ativas. A Reinicialização que consiste em modificar, atualizar e registrar uma nova configuração sobre os registros do sistema.

Tipo: String

Valor: Valor que representa a forma de manipular faltas sendo os valores as técnicas de diagnostico, isolamento, reconfiguração e reinicializarão.

4.3.4.3 Monitoramento

A subcaracterística de Monitoramento descreve as informações necessárias para realizar o monitoramento das vulnerabilidades dentro do sistema. Esse processo consiste na modelagem de vulnerabilidades, identificando possíveis problemas e ataques que podem ocorrer ao componente. Dentro desse processo de modelagem podem ocorrer ações de tratamento e classificação de novas ameaças.

4.3.4.3.1 Atributo Processo de Modelagem de Vulnerabilidades

Esse atributo possui a responsabilidade de definir o processo para a modelagem de vulnerabilidades de um componente. É um processo formal para identificar, documentar e mitigar ameaças de segurança a um sistema de software. Além disso, ajuda a times de desenvolvedores a entender a forma de trabalho de ameaças, observando o sistema a partir dos olhos de um potencial adversário (OLADIMEJI ET AL., 2006). Existem diversos processos que auxiliam essa modelagem como o Microsoft STRIDE/DREAD *model,* o *Trike*, o *Common Vulnerability Scoring System* (CVSS) e o *Operationally Critical Threat, Asset, and Vulnerability Evaluation*, OCTAVE.

Tipo: String

Valor: Valor que representa o processo de modelagem de vulnerabilidades escolhido sendo os valores o STRIDE/DREAD, TRIKE, CVSS, OCTAVE.

4.3.4.4 Considerações Finais

Para finalizar a característica de Controle de Vulnerabilidades será realizada uma descrição sucinta de cada subcaracterística com os atributos e tipos destes. O Quadro 4-7 mostra todos os itens que compõem a característica.

Para a característica de Controle de Vulnerabilidades foram definidas três subcaracterísticas: *Classificação* que define qual é o tipo da vulnerabilidade e suas origens; *Tratamento* que define qual é a forma de detectar e manipular as vulnerabilidades; *Monitoramento* que define qual é o processo de modelagem de vulnerabilidades; Dentro da subcaracterística *Classificação* foram definidos três atributos com tipos e valores:

- **Tipo**: Atributo que define o tipo da vulnerabilidade. É representado por um atributo do tipo String. O seu valor é definido como erro, falta e falha.

- **Origem de Faltas**: Atributo que define qual é a origem das faltas de um componente. É representado por um atributo do tipo String. O seu valor é definido como fase de criação, fronteiras do sistema, causa fenomenológica entre outras origens.
- **Domínio de Falhas**: Atributo que define qual é o domínio das falhas de um componente. É representado por um atributo do tipo String. O seu valor é definido como os domínios de conteúdo, tempo, conteúdo e tempo.

Quadro 4-7. Descrição da Característica Controle de Vulnerabilidade

Característica	Subcaracterística	Atributo	Tipo	Valor
Controle de Vulnerabilidades	Classificação	Tipo	String	erro, falta, falha
		Origem de Faltas	String	fase de criação, fronteiras do sistema, causa fenomenológica
		Domínio de Falhas	String	conteúdo, tempo, conteúdo e tempo
	Tratamento	Detecção	String	concorrente, preemptiva
		Manipulação de Erros	String	rollback, rollforward, compensation
		Manipulação de Faltas	String	diagnóstico, isolamento, reconfiguração, reinicialização
	Monitoramento	Processo de Modelagem de Vulnerabilidades	String	STRIDE/DREAD, TRIKE, CVSS, OCTAVE

Dentro da subcaracterística *Tratamento* foram definidos três atributos com tipos e valores:

- **Detecção:** Atributo que define qual é a forma de detecção de vulnerabilidades de um componente. É representado por um atributo do tipo String. O seu valor é definido como as formas concorrente e preemptiva.
- **Manipulação de Erros:** Atributo que define qual é a forma de manipulação de erros de um componente. É representado por um atributo do tipo String. O seu valor é definido como as formas rollback, rollforward e compensation.
- **Manipulação de Faltas:** Atributo que define qual é a forma de manipulação de faltas de um componente. É representado por um atributo do tipo String. O seu valor é definido como as formas diagnóstico, isolamento, reconfiguração, reinicialização.

Dentro da subcaracterística *Monitoramento* foi definido um atributo com tipos e valores:

- **Processo de Modelagem de Ameaças:** Atributo que define qual é a forma de modelar as ameaças de um componente. É representado por um atributo do tipo String. O seu valor é definido pelos processos STRIDE/DREAD, TRIKE, CVSS, OCTAVE.

4.3.5 Característica de Manipulação de Dados

Conforme definido a Manipulação de Dados é o processo de validação de dados fornecidos por outras entidades em um conjunto de regras pré-definidas (NETLAND ET AL., 2006). Essa manipulação dos dados de entrada é importante para evitar ataques e vulnerabilidades, pois através de um dado malicioso, podem despertar instâncias que falhas até então dormentes gerando erros na execução da funcionalidade do componente.

Ao planejar a utilização dessa característica dentro de um componente de software, existem dois itens a serem observados: o formato dos dados e a validação destes dados.

4.3.5.1 Dado

A subcaracterística de Dado descreve as informações necessárias para padronizar o formato dos dados utilizados no componente. Esse formato pode estar nos formatos de moedas, números, arquivos e a parte de internacionalização, que atualmente é normal criar um componente que pode entrar em contato com todo o mundo.

4.3.5.1.1 Atributo Formato

Esse atributo possui a responsabilidade de definir o formato de um dado de entrada no componente. Esse formato pode ser as moedas de um país, a data e hora do sistema pelo padrão ISO 8601 (I2004), a parte de interface com classes de identificação de biometria pelo padrão ISO 19794 (2011a), arquivos como documentos do Word, documentos do Excel ou pontos relacionados da linguagem de programação.

Tipo: String

Valor: Valor que representa o tipo do formato de entrada de dados como as moedas de um país, a data e hora do sistema entre outras formas de padronização.

4.3.5.1.2 Atributo Internacionalização

Esse atributo possui a responsabilidade de indicar se o componente está preparado para a utilização de caracteres de formato não padrão, como para os idiomas orientais, representando assim a internacionalização do componente (HOWARD; LEBLANC, 2002).

Tipo: String

Valor: Valor que representa o formato de internacionalização como o padrão ISO-8859-1

4.3.5.2 Validação

A subcaracterística de Validação descreve as informações necessárias para realizar a validação dos dados de entrada. Essa validação permite identificar dados estranhos ao sistema que podem ser brechas para ataques ao componente. Para realizar a validação existem diversas estratégias que vão desde manter interfaces gargalos para que todo e qualquer dado tenha que ser inserido por estas e assim centralizar a validação dos dados com uma estratégia.

4.3.5.2.1 Atributo Estratégia de Defesa

Esse atributo possui a responsabilidade de definir a estratégia de defesa utilizada para facilitar e centralizar a validação dos dados de entrada. Essa estratégia possui duas formas (HOWARD; LEBLANC, 2002). A primeira é o limite de confiança ou principio de defesa em profundidade que consiste em uma camada em que todos os dados que chegarem até ali já foram validados por outras camadas, não podendo gerar erros. A segunda forma consiste nos pontos de estrangulamento que são pontos de entrada, que controlam, a partir de validações, a entrada de dados no sistema.

Tipo: String

Valor: Valor que representa a estratégia de defesa para um componente. Pode ser representadas pelas formas de limite de confiança e pontos de estrangulamento.

4.3.5.2.2 Atributo Estratégia de Validação

Esse atributo possui a responsabilidade de definir a estratégia de validação dos dados de entrada de um componente. Existem três formas de validação de acordo com OWASP (2012): As listas brancas que são um conjunto de valores aceitos pelo componente, sendo que somente esses podem passar pela fase de validação. As listas pretas que são um conjunto de valores não aceitos pelo componente, sendo que somente valores diferentes destes é que podem passar pela fase de validação. A limpeza de dados que consiste em modificar a entrada do usuário para um formato aceito pelo componente, para isso utilizando-se de listas brancas e listas pretas.

Tipo: String

Valor: Valor que representa o nome da estratégia de validação a ser utilizada no componente com os valores de Lista Branca, Lista Preta e Limpeza de Dados.

4.3.5.2.3 Atributo Algoritmo Hash

Esse atributo define se o mecanismo de log do componente utiliza a funcionalidade de geração de código-hash sobre os arquivos de log gerados. Para verificação futura da integridade destes.

Tipo: String

Valor: Valor que representa o nome do algoritmo hash associado com a função como MD5, SHA1, SHA2.

4.3.5.2.4 Atributo Expressões Regulares

Esse atributo possui a responsabilidade de definir se o componente se beneficia da utilização de expressões regulares que são textos especiais para descrever um determinado padrão (OWASP, 2012). De acordo com Howard e Leblanc (2002) expressões regulares servem a dois propósitos principais. Encontrar dados em uma determinada String. E realizar a validação de dados. Para clarear é

necessário pensar que a string validada conter um nome de arquivo, o objetivo não é encontrar o nome do arquivo, mas determinar que o nome de arquivo seja válido.

Tipo: Booleano

Valor: Valor que representa se essa operação é realizada no componente, sendo verdadeira ou falsa.

4.3.5.3 Considerações Finais

Para finalizar a característica de Manipulação de Dados será realizada uma descrição sucinta de cada subcaracterística com os atributos e tipos destes. O Quadro 4-8 mostra todos os itens que compõem a característica.

Quadro 4-8. Descrição da Característica Manipulação de Dados

Característica	Subcaracterística	Atributo	Tipo	Valor
Manipulação de Dados	Dado	Formato	String	ISO 8601, ISO 19794, APIs de linguagens
		Internacionalização	String	ISO-8859-1
	Validação	Estratégia de Defesa	String	limite de Confiança e pontos de estrangulamento
		Estratégia de Validação	String	listas brancas, listas negras, limpeza de dados
		Algoritmo Hash	String	MD5, SHA1, SHA2
		Expressões Regulares	Booleano	verdadeiro, falso

Para a característica de Manipulação de Dados foram definidas duas subcaracterísticas: *Dado* que define o formato que o componente aceita de dados; *Validação* que define o formato de validação dos dados; Dentro da subcaracterística *Dado* foram definidos dois atributos com tipos e valores:

- **Formato**: Atributo que define o formato do dado de entrada. É representado por um atributo do tipo String. O seu valor é definido pelas normas internacionais e padrões de API.
- **Internacionalização**: Atributo que define se os dados de entrada poderão trabalhar com formatos internacionais como o Unicode. É representado por um atributo do tipo String. O seu valor é o padrão de representação desses caracteres como a ISO-8859-1.

Dentro da subcaracterística *Validação* foram definidos quatro atributos com tipos e valores:

- **Estratégia de Defesa**: Atributo que define formas de defesa dentro de um componente de software. É representado por um atributo do tipo String. O seu valor são as estratégias de limite de confiança e pontos de estrangulamento.
- **Estratégia de Validação**: Atributo que define formas de validar os dados de um componente de software. É representado por um atributo do tipo String. O seu valor são as estratégias de listas brancas, listas negras e limpeza de dados.
- **Algoritmo Hash**: Atributo que define qual é o algoritmo hash utilizado para a operação de verificação de integridade de um arquivo de log. É representado por um atributo do tipo String. O seu valor MD5, SHA1, SHA2.
- **Expressões Regulares**: Atributo que define se existe utilização das expressões regulares para a validação de Dados. É representado por um atributo do tipo Booleano. O seu valor é verdadeiro ou falso.

4.3.6 Modelo Completo

O Quadro 4-9 descreve o modelo de características de segurança para componentes de software completo, com todos os dados mostrados nas seções anteriores.

Quadro 4-9. Modelo de características de segurança para componentes de software completo – Parte I

Característica	Subcaracterística	Atributo	Tipo	Valor
Criptografia de Dados	Algoritmo	Nome	String	AES, 3DES, blowfish, Elgamal, RSA, Diffie-Hellman
		Tipo Algoritmo	String	simétricos e assimétricos
		Tamanho da Chave	Inteiro	positivo em bits
		Vida Útil	Inteiro/String	tempo/unidade de tempo
		Custo Computacional Produção	Inteiro/String	tempo/unidade de tempo
		Custo Computacional Operação	Inteiro/String	tempo/unidade de tempo
	Armazenamento Chave	Forma	String	SiRiUS, Cepheus, SNAD, EFS
Autenticação	Classe de Identificação	Algo Conhecido	String	senhas, PINs, desafios-respostas
		Algo de Posse	String	cartões magnéticos, cartões com chip, tokens
		Algo Inerente	String	biometria, forma de escrita
	Combinação de Formas	Lista de Formas	String	combinação de itens das três classes
		Robustez	String	baixo, intermediário, alto, muito alto
	Algoritmos de Funções Hash Criptográficas	Nome	String	MD5, SHA1, SHA2
		Tamanho Código-Hash	Inteiro	positivo em bits
		Tamanho do Bloco	Inteiro	positivo em bits
		Custo Computacional Produção	Inteiro/String	tempo/unidade de Tempo
Controle de Vulnerabilidades	Classificação	Tipo	String	erro, falta, falha
		Origem de Faltas	String	fase de criação, fronteiras do sistema, causa fenomenológica
		Domínio de Falhas	String	conteúdo, tempo, conteúdo e tempo
	Tratamento	Detecção	String	concorrente, preemptiva
		Manipulação de Erros	String	rollback, rollforward, compensation
		Manipulação de Faltas	String	diagnóstico, isolamento, reconfiguração, reinicialização
	Monitoramento	Processo de Modelagem de Vulnerabilidades	String	STRIDE/DREAD, TRIKE, CVSS, OCTAVE

Quadro 4-9. Modelo de características de segurança para componentes de software completo – Parte II

Característica	Subcaracterística	Atributo	Tipo	Valor
Manipulação de Dados	Dado	Formato	String	ISO 8601, ISO 19794, APIs de linguagens
		Internacionalização	String	ISO-8859-1
	Validação	Estratégia de Defesa	String	limite de Confiança e pontos de estrangulamento
		Estratégia de Validação	String	listas brancas, listas negras, limpeza de dados
		Algoritmo Hash	String	MD5, SHA1, SHA2
		Expressões Regulares	Booleano	verdadeiro, falso
Auditabilidade	Formato Log	Classe de Eventos	String	emergency, alert, information, debug
		Formato	String	Syslog
	Armazenamento	Forma	String	impresso, dispositivo WORM, storage
	Operação de Armazenamento	Algoritmo Hash	String	MD5, SHA1, SHA2
		Arquivamento de Arquivo de Log	Booleano	verdadeiro, falso
		Rotação de Arquivo de Log	Booleano	verdadeiro, falso
		Compressão de Arquivo de Log	Booleano	verdadeiro, falso
		Redução de Arquivo de Log	Booleano	verdadeiro, falso
		Visualização de Arquivo de Log	Booleano	verdadeiro, falso
	Operação de Análise	Filtragem de Eventos	Booleano	verdadeiro, falso
		Agregação de Eventos	Booleano	verdadeiro, falso
		Correlação de Eventos	Booleano	verdadeiro, falso
		Conversão de Arquivo de Log	Booleano	verdadeiro, falso
		Normalização de Arquivo de Log	Booleano	verdadeiro, falso
		Relatório de Arquivo de Log	Booleano	verdadeiro, falso

4.4 Considerações sobre o Capítulo

Este capítulo descreveu os passos necessários para chegar a estrutura do modelo proposto. Esses passos que foram as revisões sistemáticas para obter os modelos de qualidade para componentes de software existente na engenharia de software. Após essa busca esses modelos foram analisados buscando características, subcaracterísticas e atributos relacionados a área de segurança de software, do quais foram obtidas diversas áreas.

Essas áreas foram submetidas a uma extensa pesquisa buscando quais eram as necessidades para construir determinada área dentro de um componente de software. Com essas necessidades foi construído o modelo de qualidade de características internas de segurança de componentes de software, descrevendo suas características, subcaracterísticas, atributos e tipos de valores.

CAPÍTULO 5 - DISCUSSÃO DOS RESULTADOS

Na teoria, não há nenhuma diferença entre teoria e prática.
Mas na prática há.
- Jan L. A. van de Snepscheut – Professor e Cientista da Computação

Este capítulo apresenta a avaliação do modelo de qualidade de características internas de segurança de componentes de software por especialistas da área de Segurança da Informação. Esses especialistas são apresentados por seus perfis. O método de avaliação utilizado é apresentado juntamente com a tabulação de resultados. Finalmente são apresentadas as reflexões acerca dos resultados obtidos, das generalizações e da validade e confiabilidade da pesquisa.

5.1 Avaliação utilizando especialistas

A opinião de especialistas, segundo Garcia (2010), pode ser definida como uma “série de esforços científicos que são empregados para interpretar dados, prever o comportamento de um sistema e avaliar incertezas, sendo que estas especulações, suposições e estimativas servem como uma entrada cognitiva no processo de decisão”. Ainda de acordo com o autor: “Pesquisas atuais em engenharia de software têm feito constantemente o uso da opinião de especialistas”.

5.1.1 Perfis dos especialistas

Para avaliação do modelo de qualidade proposto neste trabalho foi feita uma pesquisa com especialistas em segurança da informação provenientes, tanto da área acadêmica, quanto da indústria. Foi enviado um questionário para avaliação das características de qualidade, conforme detalhado no Apêndice A. Os nomes desses especialistas, bem como as instituições a que estão vinculados, foram mantidos em sigilo, sendo identificados apenas como Especialista 1, Especialista 2 e assim por diante.

O questionário foi enviado via correio eletrônico para dez especialistas, sendo que cinco destes realizaram a avaliação. Os perfis detalhados de cada especialista serão abordados nos próximos parágrafos.

Especialista 1

O Especialista 1 é funcionário de uma empresa pública de abrangência internacional, atuando no setor de segurança da informação. Com relação à sua experiência profissional, o Especialista 1 possui 8 anos de experiência nas áreas de engenharia de requisitos, análise e projeto de sistemas e desenvolvimento de componentes de software; 5 anos de experiência nas áreas de arquitetura de sistemas, arquitetura de segurança e implementação; e, 2 anos de experiência na área de testes de software.

Na área de Ensino possui mais de 8 anos de experiência em treinamento corporativo e 2 anos de experiência na área de ensino e pesquisa acadêmica. Esse especialista já desenvolveu trabalhos acadêmicos na área de segurança de software como uma monografia de graduação, monografia de especialização e a publicação de alguns artigos na área de segurança da informação. Além de orientar trabalhos de conclusão de curso de graduação. Ele possui uma certificação da área de segurança a *Certified Secure Software Lifecycle Professional* (CSSLP) e é Auditor Líder em Sistemas de Segurança da Informação.

Especialista 2

O Especialista 2 é sócio de uma empresa de consultoria em tecnologia da informação. Na área de desenvolvimento possui mais de 8 anos de experiência nas áreas de análise e projeto de sistemas, arquitetura de sistemas, arquitetura de segurança, testes de software e implantação de sistemas; 8 anos de experiência na área de engenharia de requisitos; e, 5 anos de experiência nas áreas de desenvolvimento de componentes de software e implementação.

Na área de Ensino possui mais de 8 anos de experiência em treinamento corporativo e 5 anos de experiência na área de ensino e pesquisa acadêmica. Esse especialista já publicou alguns artigos na área de segurança da informação, além de orientar trabalhos de conclusão de curso de graduação. Ele possui as certificações *Hungarian Testing Board* (HTB), *Tenable Certified Nessus Auditor* (TCNA), *Snort*

Certified Professional (SnortCP), *Open Vulnerability Assessment System* (OpenVAS) e *IpTables Certification*.

Especialista 3

O Especialista 3 é professor de graduação e mestrado em uma universidade pública. Na área de desenvolvimento possui 2 anos de experiência na área de implementação.

Na área de Ensino possui mais de 8 anos de experiência na área de ensino e pesquisa acadêmica. Esse especialista já desenvolveu trabalhos acadêmicos na área de segurança de software como dissertação de mestrado, tese de doutorado e a publicação de alguns artigos na área de segurança da informação, além de orientar trabalhos de conclusão de curso de graduação e dissertações de mestrado.

Especialista 4

O Especialista 4 é funcionário de um grande conglomerado médico. Na área de desenvolvimento possui 2 anos de experiência nas áreas de análise e projeto de sistemas, arquitetura de sistemas, implementação e testes de software.

Na área de Ensino possui 2 anos de experiência na área de ensino e pesquisa acadêmica. Esse especialista já desenvolveu trabalhos acadêmicos na área de segurança de software como uma monografia de graduação e uma monografia de especialização.

Especialista 5

O Especialista 5 é professor de graduação em uma universidade pública. Não possui nenhuma experiência na área de desenvolvimento corporativo.

Na área de Ensino possui 2 anos de experiência em treinamento corporativo e 8 anos de experiência na área de ensino e pesquisa acadêmica. Esse especialista já desenvolveu trabalhos acadêmicos na área de segurança da informação como uma dissertação de mestrado e uma tese de doutorado. O Especialista 5 realizou publicação de alguns artigos na área de segurança da informação, além de orientar trabalhos de conclusão de curso de graduação.

5.1.2 Método de Avaliação

O principal objetivo a ser alcançado com a avaliação dos especialistas era avaliar a distribuição das características, subcaracterísticas e atributos do modelo de qualidade, bem como a sua pertinência ou não em um modelo desta natureza. Para alcançar esse objetivo foi utilizada a escala de Likert, para medir o nível de concordância/discordância em relação a cada afirmação. Para a avaliação das características, subcaracterísticas e atributos foi utilizada a escala com as seguintes opções:

- Não é Importante;
- Importância Baixa;
- Importância Intermediária;
- Importância Alta;
- Imprescindível.

Para a agregação dos valores obtidos, bem como a interpretação dos resultados, optou-se por sumarizá-los conforme ilustrado na Figura 5-1.

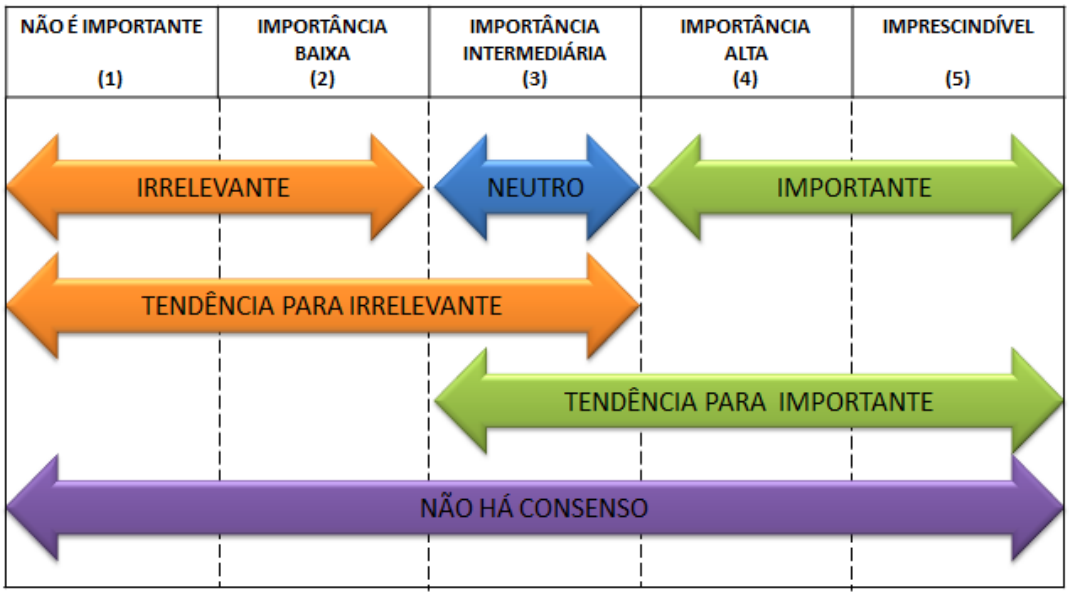

Figura 5-1. Escala para interpretação dos resultados (Fonte: O Autor).

Foi considerado que um determinado item foi avaliado pelos especialistas como *importante*, caso as respostas de todos os avaliadores tenham sido entre

“importância alta” e “imprescindível”. Neste caso, entende-se que os especialistas concordam que o item é relevante para o modelo e que, portanto, deve ser mantido.

Foi considerado que um determinado item foi avaliado pelos especialistas como *irrelevante*, caso as respostas de todos dos avaliadores tenham sido entre “não é importante” e possui “importância baixa”. Neste caso, entende-se que os especialistas não concordam que o item é relevante para o modelo e que, portanto, ele deve ser retirado do modelo.

Quando as avaliações dos especialistas incluírem opiniões classificadas como “importância intermediária”, mas ainda assim concentradas em um dos dois extremos, então se entende que estão classificados como *tendência para importante* ou *tendência para irrelevante,* conforme ilustrado na Figura 5-1. Esta interpretação também foi utilizada quando todos os votos se concentram de um dos lados, mas apenas um dos votos ficou posicionado no lado oposto.

Nos demais casos, entende-se que não houve um consenso e que o item necessita de uma melhor avaliação.

5.2 Análise dos resultados da pesquisa

A pesquisa consistiu na avaliação de diversos conceitos da área de segurança relacionados ao modelo de qualidade como as características, as subcaracterísticas e os atributos, que foram incluídos na proposta de modelo apresentada. Assim, como pôde ser observado no Apêndice A, em um primeiro momento não foi citado o modelo de qualidade em si, mas foram citados os itens da área de segurança que compuseram o modelo.

O primeiro ponto avaliado foram as características ou grandes áreas da segurança da informação. Os subitens de cada uma das áreas foram avaliados nas outras cinco questões, finalizando com a avaliação sobre a segurança de itens de autenticação.

5.2.1 Avaliação das Características do Modelo de Qualidade

A primeira questão está relacionada com a importância das grandes áreas de segurança da informação com a seguinte assinatura: *“2.1 - Em sua opinião, qual é a importância das seguintes áreas de segurança:”*. Essa questão avalia qual é a importância das características definidas no modelo, considerando a vivência e a

experiência do entrevistado. A Tabela 5-1 apresenta as respostas dadas pelos especialistas.

Tabela 5-1. Questão - Qual é a importância das seguintes áreas de segurança.

Área	Não é importante	Import. Baixa	Importância Intermediária	Import. Alta	Imprescindível	Análise
Auditabilidade	-	-	-	2	3	Importante
Autenticação	-	-	1	1	3	Tendência para Importante
Criptografia de dados	-	-	1	2	2	Tendência para Importante
Controle de vulnerabilidades	-	-	-	3	2	Importante
Manipulação de dados	-	-	1	3	1	Tendência para Importante

De acordo com a avaliação dos especialistas, todas as características mencionadas foram analisadas como *Importante* ou como tendo uma *Tendência para Importante*. Junto com essa questão foi perguntado se havia alguma outra área da segurança que não fora mencionada na questão. Os comentários estão transcritos no Quadro 5-1.

Quadro 5-1. Comentários dos Especialistas sobre Áreas de Segurança.

Comentários dos Especialistas sobre Áreas de Segurança	
Identificação	Comentário
Especialista 1	“Não há a menção da questão da segurança no processo de desenvolvimento de software, como por exemplo, a classificação e sigilo de informações a partir dos desenvolvedores ou ameaças internas referentes a pessoas”.
Especialista 4	“Tratamento das pessoas, vejo que as áreas acima mencionaram somente a tecnologia, mas as pessoas tem um papel fundamental na segurança da informação. Como a empresa trata os seus funcionários? Existem contratos assinados sobre roubo de informação? Vale a pena treinar as pessoas com relação à exposição dos dados da empresa? Também não foi mencionado como a empresa trata do seu bem ativo, o processo que está sendo aplicado é o correto? há uma melhoria contínua? normas ISO? auditoria é com frequência?”

Como se pode observar, as observações questionaram basicamente o papel do indivíduo na segurança da informação e também o papel do processo de desenvolvimento utilizado na produção/manutenção do componente de software.

O Especialista 1 e o Especialista 4 criaram observações a respeito das pessoas no processo de desenvolvimento de software. O foco deste modelo é mais técnico, ou seja, o auxílio ao desenvolvimento de novos componentes de software com mais segurança em suas características internas. O item de sigilo da informação está mais relacionado como uma parte importante de um processo corporativo de desenvolvimento de software, que não é o foco deste trabalho.

O Especialista 4 observa outro ponto relacionado aos bens ativos e aos processos que são aplicados dentro de uma corporação. Esse ponto está relacionado novamente ao processo de desenvolvimento de uma corporação, não sendo retratado neste trabalho.

5.2.2 Avaliação da Característica Auditabilidade

A segunda questão está relacionada com a importância dos itens internos da característica de Auditabilidade. A assinatura da questão é a seguinte: *"Em sua opinião, qual é a importância das seguintes características referentes à área de Auditabilidade:"*. A Tabela 5-2 mostra as respostas dadas pelos especialistas.

De acordo com a avaliação dos especialistas, a maioria das características mencionadas foram analisadas como tendo uma *Tendência para Importante* ou *Importante*.

O Atributo Classe de Eventos não foi obtido um consenso entre os especialistas. Assim este será mantido no modelo de qualidade, pois agrupa informações relacionadas com a natureza do evento, como Debug, Warning e Error, facilitando a busca e filtragem de eventos, principalmente os mais críticos ao sistema.

O Atributo Compressão de Arquivo de Log foi avaliado com uma Tendência para Irrelevante. Este atributo será mantido no modelo de qualidade, pois como sua utilização será para desenvolvedores e arquitetos com pouco conhecimento na área de segurança é necessário que eles possam conhecer as opções existentes dentro da área de auditabilidade. O mesmo acontecendo com os atributos Rotação de Log, Agregação de Eventos de Arquivo de Log, Conversão de Arquivo de Log e Normalização de Arquivo de Log.

Tabela 5-2. Questão - Qual é a importância das seguintes características referentes à área de Auditabilidade.

Característica	Não é importante	Import. Baixa	Importância Intermediária	Import. Alta	Imprescindível	Análise
Classe de eventos	1	1	-	2	1	Não há Consenso
Padrão de formato do arquivo de log	-	2	1	2	-	Neutro
Armazenamento do arquivo de log	-	1	1	3	-	Tendência para Importante
Integridade de arquivo de log	-	1	1	2	1	Tendência para Importante
Arquivamento de arquivo de log	-	1	1	3	-	Tendência para Importante
Rotação de arquivo de log	1	-	3	1	-	Tendência para Irrelevante
Compressão de arquivo de log	-	2	3	-	-	Tendência para Irrelevante
Visualização de arquivo de log	-	-	-	2	3	Importante
Filtragem de eventos de arquivo de log	-	-	3	1	1	Tendência para Importante
Agregação de eventos de arquivo de log	1	1	2	1	-	Tendência para Irrelevante
Correlação de eventos de arquivo de log	-	2	-	3	-	Tendência para Importante
Conversão de arquivo de log	-	2	2	1	-	Tendência para Irrelevante
Normalização de arquivo de log	-	3	1	1	-	Tendência para Irrelevante
Relatório de arquivo de log	-	1	-	2	2	Tendência para Importante

O Atributo Padrão de Formato de Arquivo de Log foi avaliado como neutro. Este será mantido no modelo de qualidade, pois em uma empresa de grande porte é necessária uma padronização do formato de arquivo de log para melhor controle dos auditores de sistema. Outra vantagem ao adotar uma padronização de formato é que é necessário apenas um programa para realizar a análise dos arquivos de log.

Junto com essa questão foi perguntado se havia alguma outra área da segurança que não fora mencionada na questão. Os comentários estão transcritos no Quadro 5-2.

Quadro 5-2. Comentários dos Especialistas sobre a Área de Auditabilidade.

Comentários dos Especialistas sobre a Área de Auditabilidade	
Identificação	Comentário
Especialista 1	"Classificação de acesso dos logs não foi mencionado. Ex.: Logs públicos, restritos...".

O Especialista 1 observou o fato da classificação de acesso dos arquivos de logs. Esse é um item realmente muito importante, que será incorporado ao modelo como um atributo de nome Classificação que pertence à subcaracterística Armazenamento da característica de Auditabilidade.

5.2.3 Avaliação da Característica Autenticação

A terceira questão está relacionada com a importância dos itens internos da característica de Autenticação. A assinatura da questão é a seguinte: *"Em sua opinião, qual é a importância das seguintes características referentes à área de Autenticação:"*.

A Tabela 5-3 mostra as respostas dadas pelos especialistas.

Tabela 5-3. Questão - Qual é a importância das seguintes características referentes à área de Autenticação.

Característica	Não é importante	Import. Baixa	Importância Intermediária	Import. Alta	Imprescindível	Análise
Algoritmos Hash	-	1	-	3	1	Tendência para Importante
Forma de Armazenamento	-	1	-	3	1	Tendência para Importante
Algo que você conhece (Senhas)	-	1	1	2	1	Tendência para Importante
Algo de Posse (Tokens)	-	1	-	3	1	Tendência para Importante
Algo Inerente (Digitais)	-	-	3	2	-	Tendência para Importante
Chaves Criptográficas Públicas	-	-	-	1	4	Importante

De acordo com a avaliação dos especialistas, todas as características mencionadas foram analisadas como *Importante* ou como tendo uma *Tendência para Importante*. Junto com essa questão foi perguntado se havia alguma outra área da segurança que não fora mencionada na questão. Os comentários estão transcritos no Quadro 5-3.

Quadro 5-3. Comentários dos Especialistas sobre a Área de Autenticação.

Comentários dos Especialistas sobre a Área de Autenticação	
Identificação	Comentário
Especialista 1	“Não identifiquei questões relativas a Single Sign-On, nem a integração plataforma alta/baixa para autenticação e autorização”.
Especialista 4	“Utilização de biometria e controle de acesso à informação. Quem acessou, que horas e porque?”.

O Especialista 1 mencionou o conceito de autenticação único, chamado de Single Sign-On (SSO) e a integração de autenticação entre plataformas Altas, como Cobol, Natural e plataformas Baixas como Java e Asp. De fato, ambas as observações são importantes para a integração entre componentes ou sistemas de informação. Mas como o modelo descrito neste trabalho possui um foco sobre as características internas de um componente, essa integração, que é uma característica externa não foi desenvolvida. Isso pode ser considerado como um trabalho futuro, de refinamento e ampliação do modelo.

O Especialista 4 mencionou o fator de um log para uso da autenticação utilizando biometria. Na verdade essa observação consiste em uma combinação das características de Auditabilidade e Autenticação. A biometria está contemplada no atributo Algo Inerente da característica de Autenticação. A parte de armazenamento de eventos de um usuário sobre o sistema está contemplada na utilização da característica de Auditabilidade sobre o projeto. Finalmente o item de Autenticação é contemplado ao implementar essa característica no projeto de software.

5.2.4 Avaliação da Característica Criptografia de Dados

A quarta questão está relacionada com a importância dos itens internos da característica de Criptografia de Dados. A assinatura da questão é a seguinte: *“Em sua opinião, qual é a importância das seguintes características referentes à área de Criptografia de Dados:”*. A Tabela 5-4 mostra as respostas dadas pelos especialistas.

De acordo com a avaliação dos especialistas, a maioria das características mencionadas foi analisada como *Importante* ou como tendo uma *Tendência para*

Importante. No entanto, têm-se características avaliadas como *Tendência para Irrelevante* e também como *Não há consenso.*

Tabela 5-4. Questão - Qual é a importância das seguintes características referentes à área de Criptografia de Dados.

Característica	Não é importante	Import. Baixa	Importância Intermediária	Import. Alta	Imprescindível	Análise
Algoritmos simétricos	-	1	1	1	2	Tendência para Importante
Algoritmos assimétricos	-	2	-	-	3	Não há consenso
Tamanho da chave criptográfica	-	1	-	2	2	Tendência para Importante
Vida útil da chave criptográfica	-	-	1	2	2	Tendência para Importante
Custo computacional para produzir a chave criptográfica	-	2	2	-	1	Tendência para Irrelevante
Custo computacional para o uso da chave criptográfica	-	2	2	1	-	Tendência para Irrelevante
Armazenamento da chave criptográfica	-	-	-	1	4	Importante
Verificação de integridade da chave criptográfica	-	-	-	1	4	Importante

No Atributo Algoritmos Assimétricos não foi obtido um consenso entre os especialistas. Apesar disto, 3 especialistas apontaram o item como *Imprescindível.* Assim, definiu-se que este atributo será mantido no modelo de qualidade, pois estes algoritmos possuem uma ampla utilização na criptografia de documentos e mensagens, principalmente para a internet e projetos onde não se conhece o número de usuários. O dono da chave pode distribuir a sua chave pública para que as pessoas enviem mensagens criptografadas para ele, utilizando a sua chave privada para a decifragem.

Os atributos Custo Computacional para Produzir a Chave Criptográfica e o Custo Computacional para o Uso da Chave Criptográfica foram avaliados com uma *Tendência para Irrelevante*. Nos dias atuais os custos de produção e utilização de chaves criptográficas diminuiram, principalmente pela expansão da capacidade do

hardware existente, mas ainda existem aplicações com dispositivos com pouco poder de processamento. Dessa forma, estes atributos serão mantidos no modelo de qualidade, para que possam ser avaliados em experimentos práticos, de modo que possam ser consultados em projetos de componentes para esses dispositivos.

Junto com essa questão foi perguntado se havia alguma outra área da segurança que não fora mencionada na questão. Os comentários estão transcritos no Quadro 5-4.

Quadro 5-4. Comentários dos Especialistas sobre a Área de Criptografia de Dados.

Comentários dos Especialistas sobre a Área de Criptografia de Dados	
Identificação	Comentário
Especialista 1	“Uma técnica utilizada aqui na empresa é a junção da criptografia simétrica e assimétrica no mesmo processo criptográfico. Talvez fosse interessante analisar esta possibilidade”.
Especialista 3	“O atributo de Gerência de chaves é imprescindível”.

O Especialista 1 mencionou o fator da utilização dentro de um mesmo sistema de algoritmos criptográficos simétricos e algoritmos criptográficos assimétricos dentro de um mesmo processo criptográfico. O modelo teoricamente contempla esse item, já que o arquiteto poderia escolher dois algoritmos criptográficos, um simétrico e outro assimétrico, podendo assim construir um processo híbrido de criptografia. Como exemplo poderia ser um processo que gerasse uma chave criptográfica simétrica, esta fosse cifrada por uma chave pública assimétrica do destinatário. Após o envio dessa mensagem cifrada com a chave simétrica, o destinatário poderia decifrar a mensagem obtendo assim a chave simétrica do remetente. Com isso as mensagens poderiam ser trocadas via cifragem da chave do algoritmo simétrico, caracterizando dessa forma um processo híbrido conforme o Especialista 1 mencionou.

O Especialista 3 comentou sobre o gerenciamento de chaves criptográficas. Este é um item muito importante para uma empresa que produz suas próprias chaves criptográficas. Assim este item será incorporado ao modelo como um atributo de nome Gerenciamento de Chaves dentro da subcaracterística Armazenamento da Chave da característica Criptografia de Dados.

5.2.5 Avaliação da Característica Controle de Vulnerabilidades

A quinta questão está relacionada com a importância dos itens internos da característica de Controle de Vulnerabilidades. A assinatura da questão é a seguinte: *"Em sua opinião, qual é a importância das seguintes características referentes à área de Controle de Vulnerabilidades:".* A Tabela 5-5 mostra as respostas dos especialistas.

Tabela 5-5. Questão - Qual é a importância das seguintes características referentes à área de Controle de Vulnerabilidades.

Característica	Não é importante	Import. Baixa	Importância Intermediária	Import. Alta	Imprescindível	Análise
Tipo de Falhas	-	-	1	1	3	Tendência para Importante
Origem de Falhas	-	-	1	1	3	Tendência para Importante
Detecção de Falhas	-	-	-	2	3	Importante
Manipulação de Erros	-	-	-	4	1	Importante
Processo de Modelagem de Ameaças	-	1	1	2	1	Tendência para Importante

De acordo com a avaliação dos especialistas, todas as características mencionadas foram analisadas como *Importante* ou como tendo uma *Tendência para Importante*. Junto com essa questão foi perguntado se havia alguma outra área da segurança que não fora mencionada na questão. Os comentários estão transcritos no Quadro 5-5.

Quadro 5-5. Comentários dos Especialistas sobre a Área de Controle de Vulnerabilidades.

Comentários dos Especialistas sobre a Área de Criptografia de Dados	
Identificação	Comentário
Especialista 1	"Ameaças relacionadas a pessoas não está mapeado neste questionário. Outros pontos a serem destacados são testes de segurança Caixa Branca: Análise de vulnerabilidades na análise de código; e testes de segurança Caixa Preta: Busca de vulnerabilidades, pois estes testes ajudam a encontrar vulnerabilidades quando aplicado mais profundamente".
Especialista 4	"A Auditoria de Código é fator que deve ser levado em consideração para evitar ameaças".

O Especialista 1 mencionou que ameaças relacionadas a pessoas não estão contidas no modelo. Aqui existem dois pontos a serem discutidos. O primeiro ponto é a origem da ameaça ao componente. Esta ameaça está contemplada no atributo Origem de Faltas com a ameaça humana representada como um tipo deste atributo. O segundo ponto consiste no sigilo da informação. Esse sigilo da informação está relacionado ao processo interno da empresa, evitando que os funcionários possam transmitir o conhecimento sobre inovações e patentes a outras empresas ou pessoas. Esse ponto não faz parte do escopo deste trabalho.

O Especialista 1 comentou ainda a respeito dos testes de segurança do tipo Caixa Branca (análise de código) e Caixa Preta (busca de vulnerabilidades). O teste de segurança de Caixa Branca também foi mencionado pelo Especialista 4. Esses dois itens são muito importantes para uma boa análise de vulnerabilidades de um componente de software, mas são exemplos de testes que podem ser realizados para a proteção. Assim eles serão incorporados ao modelo na parte de exemplos do atributo Detecção da subcaracterística Tratamento da característica Controle de Vulnerabilidades.

5.2.6 Avaliação da Característica Manipulação de Dados

A sexta questão está relacionada com a importância dos itens internos da característica de Manipulação de Dados. A assinatura da questão é a seguinte: *"Em sua opinião, qual é a importância das seguintes características referentes à área de Manipulação de Dados:".* A Tabela 5-6 mostra as respostas dadas pelos especialistas.

De acordo com a avaliação dos especialistas, a maioria das características mencionadas foi analisada como *Importante* ou como tendo uma *Tendência para Importante*.

Os itens Utilização de Expressões Regulares e Formato dos Dados foram avaliados com uma *Tendência para Irrelevante*. Apesar da avaliação dos especialistas de forma negativa, uma não padronização no formato de dados pode afetar, por exemplo, a estratégia de validação de um sistema, pois sem o padrão de formato, atacantes podem inserir conteúdos maliciosos dentro de arquivos e enviá-los ao sistema podendo causar falhas de segurança. As expressões regulares também entram neste sentido, pois caso um atacante possua um conhecimento

avançado sobre expressões regulares, pode extrair informações privilegiadas da base de dados. Assim é interessante manter esses atributos para alertar ao arquiteto ou desenvolvedor sobre a importância de criar padrões de formatos e expressões dentro de um componente.

Tabela 5-6. Questão - Qual é a importância das seguintes características referentes à área de Manipulação de Dados.

Característica	Não é importante	Import. Baixa	Importância Intermediária	Import. Alta	Imprescindível	Análise
Estratégia de defesa	-	-	1	2	2	Tendência para Importante
Estratégia de validação	-	-	1	2	2	Tendência para Importante
Algoritmo hash	-	-	-	4	1	Importante
Utilização de expressões regulares	-	1	4	-	-	Tendência para Irrelevante
Validação de dados internacionais (Unicode)	-	-	3	2	-	Tendência para Importante
Formato dos dados	1	1	2	1	-	Tendência para Irrelevante

Junto com essa questão foi perguntado se havia alguma outra área da segurança que não fora mencionada na questão. Os comentários estão transcritos no Quadro 5-6.

Quadro 5-6. Comentários dos Especialistas sobre a Área de Manipulação de Dados.

Comentários dos Especialistas sobre a Área de Criptografia de Dados	
Identificação	Comentário
Especialista 1	“Deve ser planejado estratégias específicas de backup e restore”.

O Especialista 1 mencionou o armazenamento dos dados, como backup e restore. Essa característica tem como objetivo tratar os dados de entrada, realizando a validação dessas entradas. Assim, um processo de armazenamento de dados poderia ser considerado uma expansão dessa característica, podendo ser tratado em um trabalho futuro.

5.2.7 Avaliação do Nível de Segurança dos Itens de Autenticação

A sétima questão está relacionada com o nível de segurança dos itens que realizam a autenticação sobre as informações de empresas ou sistemas. A

assinatura da questão é a seguinte: *"Em sua opinião, qual é o nível da segurança oferecido por cada um dos seguintes itens de segurança:".* A Tabela 5-7 mostra as respostas dadas pelos especialistas.

Tabela 5-7. Questão - Qual é o nível da segurança oferecido por cada um dos seguintes itens de segurança.

Característica	Nenhum	Nível Baixo	Nível Intermediário	Nível Alto	Muito Alto	Opinião Geral
Assinaturas e forma de escrita	1	2	2	-	-	Tendência para Irrelevante
Cartões com Chip + Senha	-	1	2	2	-	Tendência para Importante
Cartões RFID	-	-	4	1	-	Tendência para Importante
Certificados Digitais	-	-	-	5	-	Importante
Identificação por Desafio-Resposta	-	1	4	-	-	Tendência para Irrelevante
Impressões Digitais	-	-	2	2	1	Tendência para Importante
Iris	-	-	-	-	5	Importante
Passphrases	-	1	3	-	1	Tendência para Importante
Senhas	1	2	2	-	-	Tendência para Irrelevante
Senhas Únicas	2	1	1	1	-	Tendência para Irrelevante

De acordo com a avaliação dos especialistas, houve um mix de tendências sobre as características. Pode-se perceber que foram quatro itens com *Tendência para Irrelevante* e quatro itens com *Tendência para Importante.* Um dos itens foi considerado como *Importante.*

Como esta questão está relacionada mais com a parte de formas de autenticação e não incide sobre a estrutura do modelo, o importante aqui foi a aglutinação do conhecimento sobre as formas mais seguras de realizar uma autenticação. Junto com essa questão foi perguntado se havia alguma outra área da segurança que não fora mencionada na questão. Os comentários estão transcritos no Quadro 5-7.

Quadro 5-7. Comentários dos Especialistas sobre os Itens de Segurança.

Comentários dos Especialistas sobre a Área de Criptografia de Dados	
Identificação	Comentário
Especialista 1	"Dispositivos criptográficos com os HSM não foram mencionados".

5.2.8 Impressões dos Avaliadores Acerca do Modelo

Nesse ponto serão descritas as opiniões dos especialistas em questões mais gerais sobre o modelo de qualidade. Essas questões consistem em quatro pontos de análise geral como: a importância do modelo de qualidade proposto, a importância das características, a importância das subcaracterísticas e a importância dos atributos. Estas questões eram abertas, permitindo ao respondente manifestar-se livremente acerca do modelo avaliado.

A questão 3.1 é a seguinte: *"Como você avalia a importância do modelo de qualidade que está sendo proposto?".* As opiniões dos especialistas estão transcritas no Quadro 5-8:

Quadro 5-8. Comentários dos Especialistas sobre a Questão 3.1.

Comentários dos Especialistas sobre a Área de Criptografia de Dados	
Identificação	Comentário
Especialista 1	"Proposta interessante. Porém, faz-se necessário a regulamentação destas propostas via normas internas dentro da corporação para serem utilizadas.".
Especialista 2	"Baixa Importância".
Especialista 3	"O modelo é um guia útil, que pode ajudar no desenvolvimento de componentes mais seguros. Dois pontos que o modelo parece não contemplar e que são muito importantes são (1) vulnerabilidades de implementação (bugs, uso incorreto de APIs) e (2) vulnerabilidades emergentes, que derivam da interação (muitas vezes imprevista) entre componentes que, isoladamente, são seguros. Talvez esses pontos estejam fora do escopo do modelo, mas entendo que eles devem ser levados em consideração, ainda que no contexto maior do trabalho, mesmo que não sejam tratados diretamente.".
Especialista 4	"Um bom tema, mas que precisa ser melhorado, pois abordou apenas questões de tecnologia. Sabemos que SI é: Pessoas, processos e tecnologia".
Especialista 5	"Modelo relevante".

De acordo com as respostas acima, é importante observar que o modelo tem como principal cliente um arquiteto ou desenvolvedor de software que não possui um conhecimento especializado sobre segurança, realizando uma introdução mais rápida ao tema.

O Especialista 3 comentou a respeito das vulnerabilidades ocasionadas pelo mau uso da documentação existente da linguagem de programação e das vulnerabilidades ocasionadas pela integração de componentes. Ambos os pontos possuem uma dificuldade elevada para a identificação dessas vulnerabilidades. O primeiro ponto pode ser a identificação a partir de uma análise sobre o código, que contempla o atributo Processo de Modelagem de Vulnerabilidades dentro da subcaracterística Monitoramento da característica Controle de Vulnerabilidades.

O segundo ponto surge da integração que ocorre entre dois ou mais componentes de software. O objetivo deste trabalho é melhorar o nível de segurança das características internas de um componente, assim a integração de componentes, que pode ser considerada uma característica externa, não foi considerada neste momento, não consistindo no escopo deste trabalho. Mas essa integração entre componentes pode ser considerada um trabalho futuro de refinamento e ampliação do modelo de qualidade.

O Especialista 4 comenta que devem ser consideradas para a segurança de um sistema de informação além da tecnologia, as pessoas e os processos. O foco deste trabalho consiste na criação de um modelo de qualidade para componentes de software, ou seja, somente sobre a tecnologia. Assim estão fora do escopo os itens de segurança relacionados a pessoas e processos.

A questão 3.2 é a seguinte: *“Em sua opinião, as características levantadas no modelo são importantes para a criação de componentes de software com uma maior segurança interna? Por quê? Caso mais alguma característica seja importante, qual seria essa e o porquê dessa importância?”* As opiniões dos especialistas estão transcritas no Quadro 5-9:

Quadro 5-9. Comentários dos Especialistas sobre a Questão 3.2.

Comentários dos Especialistas sobre a Área de Criptografia de Dados	
Identificação	Comentário
Especialista 1	“Um importante item na questão de componentes de software é a publicação e versionamento. Não identificados neste processo.”.
Especialista 2	“Sim são importantes, pois devido ao baixo nível de análise e troca de informações com profissionais de segurança, administradores e análise de documentação da BUGTRAP.”.
Especialista 3	“As características levantadas são importantes, pois cobrem boa parte do espectro de preocupações de segurança de desenvolvedores. Senti falta de características de controle de acesso, isto é, a quem serão atribuídas permissões (a usuários individuais, a grupos de usuários, a funções/papeis, a níveis, etc.), e como essas permissões serão usadas para verificar se acessos são ou não autorizados. Frequentemente, existe uma "reinvenção da roda" nessa característica.”.
Especialista 4	“Sim, pois aborda muitas questões que ficam do lado interno da segurança.”.
Especialista 5	“Não tenho sugestão”.

A maioria dos especialistas relatou que as características contempladas no modelo são importantes para a troca de informações entre desenvolvedores e especialistas em segurança.

O Especialista 1 comentou sobre a publicação e o versionamento de componentes e seus arquivos. Ambos os itens são importantes para recuperar ou identificar erros em códigos antigos. Mas ambos são pontos de um processo de publicação de software de uma empresa, não contemplando o escopo deste trabalho.

O Especialista 3 questionou o fator de permissões de controle de acesso dentro do componente, como por exemplo, papéis, níveis, funções, entre outros. Esse ponto é importante para verificar o nível de autorização dentro de pontos específicos do componente, mas como a criação de papéis, níveis e funções muitas vezes é muito específica para cada projeto de componente, é muito difícil mensurar algo que funcione bem para a maioria. Esse fator de permissões pode ser

trabalhado futuramente com um maior refinamento para ser integrado ao modelo de qualidade proposto.

A questão 3.3 é a seguinte: *"Em sua opinião, as subcaracterísticas do modelo estão com uma boa aglutinação? Existem outras subcaracterísticas que poderiam estar no modelo? Existe alguma subcaracterística que poderia se tornar uma característica?"*. As opiniões dos especialistas estão transcritas no Quadro 5-10:

Quadro 5-10. Comentários dos Especialistas sobre a Questão 3.3.

Comentários dos Especialistas sobre a Área de Criptografia de Dados	
Identificação	Comentário
Especialista 1	"Não identifiquei a classificação da Informação (Confidencial, secreto, público...)".
Especialista 2	"Sim, melhorar a especificação de quais os sistemas operacionais. Na minha análise está apenas direcionado para os sistemas Microsoft.".
Especialista 3	"No geral, as subcaracterísticas parecem adequadas, embora a questão de gerência de chaves esteja um pouco superficial, e é um problema que tem que ser resolvido por qualquer aplicação que dependa de chaves criptográficas.".
Especialista 4	"Não creio".
Especialista 5	"Não tenho sugestão".

Alguns especialistas relataram que as subcaracterísticas contempladas no modelo são importantes.

O Especialista 1 comentou a respeito da falta de uma subcaracterística que tratasse a classificação da informação. Provavelmente ele esteja se referindo ao comentário realizado por ele na característica de Auditabilidade. Esse é um item realmente muito importante que será incorporado ao modelo como um atributo de nome Classificação que pertence a subcaracterística Armazenamento da característica de Auditabilidade, conforme citado anteriormente.

O Especialista 2 comentou sobre o fator da especificação de sistemas operacionais, mencionando que o modelo está direcionado para sistemas Microsoft. Como este é um modelo para descrição de características internas de componentes de software, não existe um direcionamento para sistemas operacionais ou

tecnologias específicas. O conteúdo do modelo, assim como deste trabalho, está baseado na construção de componentes.

O Especialista 3 comentou sobre o gerenciamento de chaves criptográficas. Este é um item muito importante para uma empresa que produz suas próprias chaves criptográficas. Assim este item será incorporado ao modelo como um atributo de nome Gerenciamento de Chaves dentro da subcaracterística Armazenamento da Chave da característica Criptografia de Dados.

A questão 3.4 é a seguinte: *"Em sua opinião, os atributos do modelo estão com uma boa aglutinação? Existe algum atributo que não se encontra no modelo? Existe algum atributo que pode ser excluído do modelo? Qual é a utilidade desses atributos no dia a dia de um projeto de software?"* As opiniões dos especialistas estão transcritas no Quadro 5-11:

Quadro 5-11. Comentários dos Especialistas sobre a Questão 3.4.

Comentários dos Especialistas sobre a Área de Criptografia de Dados	
Identificação	Comentário
Especialista 1	"Não identifiquei dois itens que considero muito importante neste processo: testes de segurança Caixa Branca: Análise de vulnerabilidades na análise de código; testes de segurança Caixa Preta: Scan de vulnerabilidades.".
Especialista 2	"Evitar falhas, acesso indevido de informações empresariais.".
Especialista 3	"Difícil responder sem uma análise mais aprofundada, mas a lista de atributos parece de razoável a boa (dada a natureza subjetiva da avaliação, uma lista perfeita seria praticamente impossível).".
Especialista 4	"Os atributos tem boa aglutinação sim. atributos não são excludentes e não precisam ser excluídos. os atributos tem utilidade mediana.".
Especialista 5	"Não tenho sugestão".

Alguns especialistas relataram que os atributos contemplados no modelo possuem uma boa aglutinação. O Especialista 1 comentou a respeito dos testes de segurança Caixa Branca e Caixa Preta. Esses dois itens são muito importantes para uma boa análise de vulnerabilidades de um componente de software, mas são exemplos de testes que podem ser realizados para a proteção. Assim eles serão

incorporados ao modelo na parte de exemplos do atributo Detecção da subcaracterística Tratamento da característica Controle de Vulnerabilidades.

O Especialista 2 comentou a respeito de dois itens: Evitar falhas e acesso indevido a informações. O primeiro ponto está contemplado na união das subcaracterísticas Tratamento e Monitoramento da característica Controle de Vulnerabilidades, dessa forma não seria necessário criar um item somente para tratar deste ponto.

O segundo ponto é relacionado ao acesso indevido de informações empresariais. Provavelmente o especialista está retratando aqui a preocupação sobre as informações sigilosas de um projeto, como a parte de patentes ou inovações sobre os concorrentes. Como este ponto é relacionado mais a sigilo da informação, que é um processo que uma empresa deve ter, este não se encontra no escopo deste trabalho.

5.2.9 Sumário das Análises

A partir das análises individuais das questões, foi produzido um quadro geral, que sumariza a opinião dos especialistas acerca do modelo avaliado. Para tanto foi calculada a tendência geral do modelo com o somatório de todas as questões, conforme apresentado na Tabela 5-8.

Tabela 5-8. Questão – Resultado Geral da Pesquisa.

Item	Não há Consenso	Irrelevante	Tendência para Irrelevante	Neutro	Tendência para Importante	Importante	Análise
Áreas de Segurança	-	-	-	-	3	2	Importante
Auditabilidade	1	-	5	1	6	1	Não há Consenso
Autenticação	-	-	-	-	5	1	Importante
Criptografia de Dados	1	-	2	-	3	2	Não há Consenso
Controle de Vulnerabilidades	-	-	-	-	3	2	Importante
Manipulação de Dados	-	-	2	-	3	1	Não há Consenso

De acordo com a tabela de resultado geral da pesquisa, é interessante notar que muitos atributos foram considerados com uma *Tendência Irrelevante* pelos especialistas. Esses podem ser caracterizados como operações auxiliares, não devendo ser retirados do modelo. Esses atributos são os seguintes: Rotação de

arquivo de log; Compressão de arquivo de log; Agregação de eventos de arquivo de log; Conversão de arquivo de log; Normalização de arquivo de log; Custo computacional para produzir a chave criptográfica; Custo computacional para o uso da chave criptográfica; Utilização de Expressões Regulares; e Formato dos Dados.

Em relação aos atributos que os especialistas não tiveram um consenso sobre as avaliações, é importante citar que estes são relevantes dentro de suas características, de acordo com a visão do autor.

Nos outros pontos de análise, 32 itens do modelo foram caracterizados como *Importante* ou com *Tendência para Importante*, caracterizando uma boa quantidade de itens com um alto valor de importância. Existem algumas sugestões realizadas pelos especialistas que foram acatadas para acrescentar itens importantes no modelo de qualidade, conforme explicitado nas seções anteriores.

5.3 Alterações propostas pelos especialistas

Algumas observações realizadas pelos especialistas foram consideradas para serem acrescentadas ao modelo. Assim, essa seção resume esses pontos acrescentados. Outro ponto positivo está na avaliação geral dos itens propostos no modelo, pois nenhum recebeu uma qualificação de *Irrelevante*, apenas existiram alguns itens classificados como *Tendência para ser Irrelevante*. O Quadro 5-12 apresenta o modelo de qualidade já com os novos atributos sugeridos pelos especialistas. Estes aparecem em negrito e sublinhado.

Ocorreu a inclusão do atributo Gerenciamento de Chaves na subcaracterística Armazenamento da Chave, característica Criptografia de Dados. Esse define a forma de gerenciamento das chaves dentro de um componente.

O atributo Classificação dentro da subcaracterística de Armazenamento dentro da característica Auditabilidade foi outra alteração advinda das sugestões dos especialistas. Esse atributo define a forma de proteção aos arquivos de log, impedindo que pessoas não autorizadas possam acessar esses arquivos.

Novos exemplos para o atributo Detecção da subcaracterística Tratamento da característica Controle de Vulnerabilidades foi outra melhoria incorporada. Esses dois exemplos consistem em testes de segurança em dois formatos: de Caixa Branca, na análise de código, e de Caixa Preta, na busca de vulnerabilidades.

Quadro 5-12. Modelo de características de segurança para componentes de software completo – Parte I

Característica	Subcaracterística	Atributo	Tipo	Valor
Criptografia de Dados	Algoritmo	Nome	String	AES, 3DES, blowfish, Elgamal, RSA, Diffie-Hellman
		Tipo Algoritmo	String	simétricos e assimétricos
		Tamanho da Chave	Inteiro	positivo em bits
		Vida Útil	Inteiro/String	tempo/unidade de tempo
		Custo Computacional Produção	Inteiro/String	tempo/unidade de tempo
		Custo Computacional Operação	Inteiro/String	tempo/unidade de tempo
	Armazenamento Chave	Forma	String	SiRiUS, Cepheus, SNAD, EFS
		Gerenciamento de Chaves	**String**	**Frameworks de gerenciamento de chaves**
Autenticação	Classe de Identificação	Algo Conhecido	String	senhas, PINs, desafios-respostas
		Algo de Posse	String	cartões magnéticos, cartões com chip, tokens
		Algo Inerente	String	biometria, forma de escrita
	Combinação de Formas	Lista de Formas	String	combinação de itens das três classes
		Robustez	String	baixo, intermediário, alto, muito alto
	Algoritmos de Funções Hash Criptográficas	Nome	String	MD5, SHA1, SHA2
		Tamanho Código-Hash	Inteiro	positivo em bits
		Tamanho do Bloco	Inteiro	positivo em bits
		Custo Computacional Produção	Inteiro/String	tempo/unidade de Tempo
Controle de Vulnerabilidades	Classificação	Tipo	String	erro, falta, falha
		Origem de Faltas	String	fase de criação, fronteiras do sistema, causa fenomenológica
		Domínio de Falhas	String	conteúdo, tempo, conteúdo e tempo
	Tratamento	Detecção	String	concorrente, preemptiva, **teste caixa branca, teste caixa preta**
		Manipulação de Erros	String	rollback, rollforward, compensation
		Manipulação de Faltas	String	diagnóstico, isolamento, reconfiguração, reinicialização
	Monitoramento	Processo de Modelagem de Vulnerabilidades	String	STRIDE/DREAD, TRIKE, CVSS, OCTAVE

Quadro 5-12. Modelo de características de segurança para componentes de software completo – Parte II

Característica	Subcaracterística	Atributo	Tipo	Valor
Manipulação de Dados	Dado	Formato	String	ISO 8601, ISO 19794, APIs de linguagens
		Internacionalização	String	ISO-8859-1
	Validação	Estratégia de Defesa	String	limite de Confiança e pontos de estrangulamento
		Estratégia de Validação	String	listas brancas, listas negras, limpeza de dados
		Algoritmo Hash	String	MD5, SHA1, SHA2
		Expressões Regulares	Booleano	verdadeiro, falso
Auditabilidade	Formato Log	Classe de Eventos	String	emergency, alert, information, debug
		Formato	String	Syslog
	Armazenamento	Forma	String	impresso, dispositivo WORM, storage
		Classificação	**String**	**Publico, Confidencial**
	Operação de Armazenamento	Algoritmo Hash	String	MD5, SHA1, SHA2
		Arquivamento de Arquivo de Log	Booleano	verdadeiro, falso
		Rotação de Arquivo de Log	Booleano	verdadeiro, falso
		Compressão de Arquivo de Log	Booleano	verdadeiro, falso
		Redução de Arquivo de Log	Booleano	verdadeiro, falso
		Visualização de Arquivo de Log	Booleano	verdadeiro, falso
	Operação de Análise	Filtragem de Eventos	Booleano	verdadeiro, falso
		Agregação de Eventos	Booleano	verdadeiro, falso
		Correlação de Eventos	Booleano	verdadeiro, falso
		Conversão de Arquivo de Log	Booleano	verdadeiro, falso
		Normalização de Arquivo de Log	Booleano	verdadeiro, falso
		Relatório de Arquivo de Log	Booleano	verdadeiro, falso

5.4 Reflexões acerca dos resultados obtidos

Conforme estabelecido no capítulo 1, os objetivos específicos são os seguintes:

(I) Identificar as grandes áreas de segurança para software e extrair as características principais para a construção do modelo;

(II) Construir um modelo de qualidade para características internas de segurança em componentes de software; e

(III) Avaliar o modelo de qualidade proposto.

Para atender ao primeiro objetivo foram definidas as grandes áreas de segurança, a partir dos itens obtidos com a revisão de literatura. Dessa definição foi realizada uma busca de materiais da área de segurança, destacando itens necessários para a correta implantação de cada área. Esses itens foram avaliados com um especialista em segurança, que indicou novos itens e materiais para ajudar nessa busca. Dessa forma o objetivo foi concluído.

Para atender ao segundo objetivo foi necessária a tabulação do modelo, a partir dos itens levantados pelo objetivo anterior. A partir destes, o modelo pode ser criado com suas características, subcaracterísticas, atributos e tipos. Essa construção está descrita na parte final do Capítulo 4. Com o modelo construído, atende-se a esse objetivo.

O último objetivo foi atendido pela análise dos dados da pesquisa feita junto a especialistas, para avaliar o modelo proposto. As conclusões levaram a um refinamento do modelo. Portanto, a partir das etapas metodológicas realizadas, foi possível atender ao objetivo geral deste trabalho que era:

Propor um modelo de qualidade para características internas de segurança de componentes de software.

Com isso, se pode responder positivamente a questão da pesquisa, que é:

É possível definir as características internas de segurança de um componente de software?

O modelo proposto, detalhado no Capitulo 4, se apresenta como uma alternativa para desenvolvedores e arquitetos que precisam melhorar a parte de segurança dentro de um componente de software. Com a definição das 5 características e de todas as subcaracterísticas e seus atributos internos, pode-se dizer que o objetivo geral da pesquisa foi alcançado com sucesso.

5.5 Reflexões acerca das generalizações

Apesar do pequeno número de especialistas que responderam o questionário durante o período de avaliação, pode se afirmar que as respostas representam uma realidade que pode ser utilizada como base para as definições apresentadas como resultado da pesquisa, dada a experiência dos envolvidos.

Essa pesquisa foi respondida por especialistas que trabalham na área ou pesquisam sobre esse tema, os quais acrescentaram alguns pontos importantes para o modelo, sendo que sua opinião foi útil para considerar o modelo com um bom refinamento e que pode ajudar a desenvolvedores e arquitetos a definir melhor itens de segurança dentro de seus softwares.

5.6 Considerações sobre o Capítulo

Este capítulo mostrou a pesquisa feita com especialistas para avaliação do modelo proposto. Inicialmente foram mostrados os perfis dos especialistas selecionados e os métodos aplicados. Em seguida foram detalhados os resultados da pesquisa, individualmente por característica. Adicionalmente foram discutidas as alterações propostas pelos especialistas. Finalmente, são apresentadas as reflexões acerca dos resultados obtidos e reflexões acerca das generalizações.

CAPÍTULO 6 - CONSIDERAÇÕES FINAIS

O futuro têm muitos nomes. Para os incapazes o inalcançável, para os medrosos o desconhecido, para os valentes a oportunidade.
- Victor Hugo – Escritor e Estadista Francês

A conclusão deste trabalho será apresentada a seguir juntamente com a relevância deste estudo, as suas principais contribuições, as suas limitações e os trabalhos futuros.

6.1 Relevância do estudo

A geração de um modelo de qualidade para características internas de componentes de software representa uma contribuição importante para a área, uma vez que não foi possível encontrar nenhum outro modelo que cobrisse os aspectos abordados por este trabalho. Foi possível apresentar um modelo que pode ser utilizado por desenvolvedores e arquitetos de software, para melhorar a questão da segurança em seus componentes e softwares.

Alguns modelos apresentados na literatura fornecem propostas para a melhoria da qualidade como um todo de componentes de software, mas nenhum deles aprofunda o conhecimento sobre segurança de software, como proposto nesse trabalho.

O modelo sintetiza o conhecimento das áreas de segurança, que muitas vezes fica restrito a uma pequena parcela dos desenvolvedores que trabalham ou se especializaram em segurança de software.

6.2 Contribuições da pesquisa

A principal contribuição deste trabalho foi a construção do modelo de qualidade para características internas de segurança de componentes de software, o qual teve como intuito mostrar a importância de criar componentes de software mais seguros, mostrando que as áreas de segurança não são tão complexas

principalmente para desenvolvedores e arquitetos não familiarizados com essa vertente. Dessa forma podem ser ampliadas as discussões sobre modelos de qualidade para componentes de software, incentivando o estudo de novas características para segurança de software.

6.3 Limitações da pesquisa

Esta pesquisa tem a abrangência de descrever as características mais importantes da segurança de software. Assim é possível que novos detalhes sobre as áreas citadas no modelo, ou até mesmo novas áreas apareçam, mas não irão alterar a estrutura básica do modelo de qualidade proposto nesse trabalho.

Outra limitação está na falta de métricas para definir melhor como avaliar cada atributo proposto, justamente pela unicidade existente em cada projeto de software.

Outro ponto a ser destacado está na quantidade de especialistas que avaliaram o modelo, pois quanto maior fosse esse número, mais opiniões e ajustes poderiam ser realizados sobre o modelo proposto. No entanto, devido à experiência dos especialistas entrevistados, é possível considerar que houve uma boa representatividade.

6.4 Trabalhos futuros

Existem diversas atividades que podem ser realizadas tendo em vista a ampliação e aperfeiçoamento deste trabalho. Um primeiro ponto é a aplicação deste modelo de qualidade em projetos de componentes de software, visando refinar os atributos do modelo, assim como realizar uma avaliação sobre estes.

Outro ponto pode estar relacionado com a inserção de novos atributos sobre o modelo de qualidade. Alguns desses atributos foram apontados pelos especialistas durante o questionário realizado como a integração de componentes e sua forma única de autenticação, no formato Single Sign-On, por exemplo. Ou atributos relacionados ao armazenamento de dados obtidos e processados pelos componentes de software como a parte de backup e restore. Ainda podem ser aplicados atributos relacionados ao controle de acesso, invocando a parte de papéis e permissões dentro das funcionalidades de um componente de software.

Um terceiro ponto de estudos futuros está relacionado com a composição de um processo de certificação de componentes focado nos aspectos relacionados à segurança de componentes;

Um quarto ponto está na criação de novos modelos de qualidade para outras características de componentes de software como usabilidade, manutenibilidade, mas com um foco de aprofundar melhor essas características. Ao final realizar uma união destes modelos em um único muito mais profundo e passível de consulta para desenvolvedores e arquitetos.

Por fim um último ponto seria a criação de uma ferramenta para apoiar o uso do modelo de qualidade apresentado nesta pesquisa, facilitando assim a tomada de decisões de um arquiteto ou desenvolvedor.

REFERÊNCIAS BIBLIOGRÁFICAS

(ALMEIDA ET AL., 2007) ALMEIDA, E. S.; ÁLVARO, A.; GARCIA, V. C.; MASCENA, J. C. C. P.; BURÉGIO, V. A. A.; NASCIMENTO, L. M.; LUCRÉDIO, D.; MEIRA. S. L. R. **C.R.U.I.S.E - Component Reuse in Software Engineering**. Recife: Gráfica Dom Bosco, 2007, 199 p.

(ALVARO, 2009) ALVARO, A. **A Software Component Quality Framework**, 2009. 223 p. Tese (Doutorado) – Pós-Graduação em Ciência da Computação, Universidade Federal de Pernambuco, Recife, 2009

(ALVARO ET AL., 2010) ALVARO, A.; ALMEIDA, E.S.; MEIRA, S.R.L. **A software component quality framework**, ACM SIGSOFT Software Engineering Notes, v.35, i.1, p.1-18, novembro, 2010.

(ANDREOU; TZIAKOURIS, 2007) ANDREOU, A. S.; TZIAKOURIS, M. **A quality framework for developing and evaluating original software components.** Information and Software Technology, v. 49, n. 2, p. 122-141, maio 2006.

(APPSIC, 2006) MEMBERS OF THE APPLICATION SECURITY INDUSTRY CONSORTIUM (AppSIC) **What Software Security Means to Business.** Datenschutz und Datensicherheit – DuD, v. 30, n. 10, p. 632-635, outubro, 2006.

(AVIZIENIS ET AL., 2004) AVIZIENIS, A.; LAPRIE, J. C.; RANDELL, B.; LANDWEHR, C. **Basic Concepts and Taxonomy of Dependable and Secure Computing**; IEEE Transactions on Dependable and Secure Computing; v. 1, i. 1, p. 11-33, janeiro, 2004.

(BAYUK, 2000) BAYUK, J. L. **Information Security Metrics: An Audited-based Approach.** IN: NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY (NIST) AND COMPUTER SYSTEM SECURITY AND PRIVACY ADVISORY BOARD (CSSPAB) WORKSHOP, 1., 2000, Washington, DC, **Anais…**, 2000.

(BASILI; ROMBACH, 1991) BASILI, V. R.; ROMBACH, H. D. **Support for Comprehensive Reuse**. Software Engineering Journal, v. 6, n.5, p. 306-316, março 1991.

(BASS ET AL., 2001) BASS, L.; BUHMAN, C.; COMELLA-DORDA, S.; LONG, F.; ROBERT, J.; SEACORD, R.; WALLNAU, K. **Volume I: Market Assessment of Component-Based Software Engineering** (CMU/SEI-2001-TN-007). Pittsburgh: Software Engineering Institute, 2001, 41p.

(BATISTA, 2007) BATISTA, C. F. A. **Métricas de Segurança de Software**, 2007, 102 p. Dissertação (Mestrado) - Programa de Pós-Graduação em Informática, Pontificia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2007

(BERTOA; VALLECILLO, 2002) BERTOA, M. F.; VALLECILLO, A. **Quality Attributes for COTS Components** In: 6TH INTERNATIONAL WORKSHOP ON QUANTITATIVE APPROACHES IN OBJECT-ORIENTED SOFTWARE ENGINEERING, 1., 2002, Malaga, ES, **Anais…**, 2002, p. 1-11

(BOEHM; BROWN; LIPOW, 1976) BOEHM, B. W.; BROWN, J. R.; ; LIPOW, M. **Quantitative evaluation of software quality.** In: ICSE '76 PROCEEDINGS OF THE 2ND INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 1., 1976, São Francisco, CA, **Anais…**, 1976, p. 592-605, 1976

(BROOKS, 1987) BROOKS, F. P. J. **No silver bullet essence and accidents of Software Engineering.** Computer, v. 20, n. 4, p. 10-19, abril, 1987

(BROWN; SHORT, 1997) BROWN, A. W.; SHORT, K.; **On components and objects the foundations of component-based development.** In: 5TH INTERNATIONAL SYMPOSIUM ON ASSESSMENT OF SOFTWARE TOOLS, 1., 1997, Washington, DC, **Anais…**, 1997, p. 112-121

(BROWN; WALLNAU, 1998) BROWN, A. W.; WALLNAU, K. C.; **The Current State of CBSE**. IEEE Software, v. 15, n. 5, p. 37-46, setembro, 1998.

(CALLAS, 2006) CALLAS, J. **An Introdution to Cryptography**. 1. ed. Estados Unidos da América:PGP Corporation, 2006. 83 p.

(CAMPADELLO; MACLAVERTY; SARIDAKIS, 2004) CAMPADELLO, S.; MACLAVERTY, R.; SARIDAKIS, T. **Security and Reliability Challenges in Component-Based Software for Consumer Devices.** IN: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING AND APPLICATIONS (IASTED), 1., 2004, Innsbruck, Austria, **Anais…**, 2004, p. 692-700

(CAPRETZ; CAPRETZ; LI, 2001) CAPRETZ, L.F.; CAPRETZ, M.A.M.; LI, D. **Component-Based Software Development**. In: ANNUAL CONFERENCE OF THE IEEE INDUSTRIAL ELECTRONICS SOCIETY (IECON), 3., 2001, Denver, CO, **Anais…**, 2001, p. 1834-1837

(CERVO, 2002) CERVO, A. L.; **Metodologia Científica**. 5. ed. Editora Pearson Prentice Hall. São Paulo, 2002.

(CHEN; ZHOU, 2010) CHEN, L.; ZHOU, S.; **The comparisons between public key and symmetric key cryptography in protecting storage systems**. IN: International Conference on Computer Application and System Modeling (ICCASM), 4. 2010, Taiyuan, China, **Anais…**, 2010, p. 494-502

(CHOWDHURY; ZULKERNINE, 2011) CHOWDHURY, I.; ZULKERNINE M. **Using complexity, coupling, and cohesion metrics as early indicators of vulnerabilities.** Journal of Systems Architecture, v. 57, n. 3, p. 294-313, março, 2011

(CHOI ET AL., 2008) CHOI, Y.; LEE, S.; SONG, H; PARK, J.; KIM, S. **Practical S/W Component Quality Evaluation Model.** In: 10TH IEEE INTERNATIONAL CONFERENCE ON ADVANCED COMMUNICATION TECHNOLOGY (ICACT 2008), 1., 2008, New York, NY, **Anais...**, 2008, p. 259-264.

(COLOMBO ET AL., 2007) COLOMBO, R. M. T.; GUERRA, A. G.; AGUAYO, M. T. V.; PERES, D. R. **Modelo de qualidade de componentes de software.** In: INTERNATIONAL ASSOCIATION FOR DEVELOPMENT OF THE INFORMATION SOCIETY (IADIS) - IBERO AMERICANA CONFERENCE WWW/INTERNET 2007, 1., 2007, Vila Real, PT, **Anais...**, 2007, p. 223-230.

(COMMONS VALIDATOR, 2012) **COMMONS VALIDATOR**, Disponível em: <http://commons.apache.org/validator> Acesso em 20 de fevereiro de 2012.

(CVSS, 2012) **Common Vulnerability Scoring System**. Disponível em: <http://www.first.org/cvss> Acesso 14 de abril de 2012.

(DEMICHIEL, 2007) DEMICHIEL, L.G. **Enterprise Java Beans (EJB) Specification**, Version 3.0, Oracle, 2007

(DETIENNE, 1991) DÉTIENNE, F. **Reasoning from a schema and from an analog in software code reuse**. In: EMPIRICAL STUDIES OF PROGRAMMERS: FOURTH WORKSHOP, 12., 1991, New Brunswick, NJ.

(DIFFIE; HELLMAN, 1976) DIFFIE, W.; HELLMAN, M. E.; **New directions in cryptography.** IEEE Transactions on Information Theory, v.22, n. 6, p. 644–654, novembro, 1976.

(DIJKSTRA, 1968) DIJKSTRA, E. W. **Letters to the editor: go to statement considered harmful**, Communications of the ACM, v.11 n.3, p.147-148, março 1968

(DOBBERTIN ET AL., 1996) DOBBERTIN, H.; BOSSELAERS, A.; PRENEEL, B.; **RIPEMD-160: A strengthened version of RIPEMD**, Lecture Notes in Computer Science, v. 1039/1996, p. 71-82, 1996.

(DOD, 1985) DEPARTMENT OF DEFENSE (DOD) **Trusted Computer System Evaluation Criteria (TCSEC)**. DOD, 1985

(DREAD, 2012) **The DREAD model**. Disponível em: <http://msdn.microsoft.com/en-us/library/ff648644.aspx> Acesso em 13 de abril de 2012.

(DROMEY, 1995) DROMEY, G. **A Model for Software Product Quality.** IEEE Transactions on Software Engineering, v. 21, n. 2, p. 146-162, fevereiro 1995.

(D'SOUZA; WILLS, 1999) D'SOUZA, D. F.; WILLS, A. C. **Objects, Components, and Frameworks with UML, The Catalysis Approach**, Addison-Wesley, USA, 1999

(EZRAN; MORISIO; TULLY, 2002) EZRAN, M.; MORISIO, M.; TULLY, C. **Practical Software Reuse**. London: Springer-Verlag, 2002, 222 p.

(FERGUSON; SCHNEIER; KOHNO, 2010) FERGUSON, N.; SCHNEIER, B.; KOHNO, T.; **Cryptography Engineering: Design Principles and Practical Applications**, Wiley, USA, 2010

(FRAKES; ISODA, 1994) FRAKES, W.; ISODA, S. **Success Factors of Systematic Reuse**. IEEE Software, v.12, n.1, p.15-19, setembro 1994.

(FREEMAN, 1983) FREEMAN, P. **Reusable software engineering: Concepts and research directions**. In WORKSHOP ON REUSABILITY IN PROGRAMMING. 9., 1983, Newport, RI. ITT Programming, Stratford, Corm., pp. 2-16.

(GAMAL, 1984) GAMAL, T. E.; A public key cryptosystem and a signature scheme based on discrete logarithms. IN: Advances in Cryptology, Proceedings of CRYPTO ’84, Santa Barbara, CA, p. 10–18, agosto, 1984.

(GARCIA, 2010) GARCIA, V. C.; **RiSE reference model for software reuse adoption in brazilian companies**. Tese doutorado – Universidade Federal de Pernambuco, CIn. Ciência da computação, 2010.

(GARVIN, 1984) GARVIN, D. A. **What Does "Product Quality" Really Mean?**. Sloan Management Review, v. 26, n. 1, outubro 1984, pp. 25-43

(GIL, 2002) GIL, A. C. **Como elaborar projetos de pesquisa**. 4. ed. São Paulo:Atlas, 2002. 175 p.

(GHOSH; MCGRAW, 1998) GHOSH, A. K.; MCGRAW, G. **An Approach for Certifying Security in Software Components**. IN: 21ST NATIONAL INFORMATION SYSTEMS SECURITY CONFERENCE, 1., 1998, Arlington, VA, **Anais…**, 1998, p. 1-7.

(GOLLMANN, 2006) GOLLMANN, D. **Computer Security**. 2. ed. West Sussex:John Wiley & Sons Ltd, 2006. 359 p.

(HEINEMAN; COUNCILL, 2001) HEINEMAN, G.; COUNCILL, W. T. **Component-Based Software Engineering: Putting the Pieces Together**, 1. Ed. Boston: Addison-Wesley, 2001.

(HENNING, 2001) HENNING, R **Information System Security Attribute Quantification or Ordering (Commonly but improperly known as .Security Metrics.)**. IN: PROCEEDINGS OF THE 1TH WORKSHOP ON INFORMATION-SECURITY-SYSTEM SCORING AND RANKING, 1., 2001, Williamsburg, VA, USA, 2001, 70 p.

(HILDERING, 2003) HILDERING, G. H. **A Graphical Modeling Language for Specifying Concurrency based on CSP**. IEEE Proceedings Software, v. 150, n. 2, p. 108-120, 2003

(HOWARD; LEBLANC, 2002) HOWARD; M.; LEBLANC, D. **Writing Secure Code**, 2. Ed. Redmont: Microsoft Press, 2002.

(IEEE, 1990) **IEEE Standard Glossary of Software Engineering Terminology**, IEEE standard 610.12, 1990

(IETF, 1992) Internet Engineering Task Force (IETF), **The MD5 Message-Digest Algorithm**, RFC 1321, abril, 1992, 22 p.

(IETF, 2001) Internet Engineering Task Force (IETF), **The BSD Syslog Protocol**, RFC 3164, agosto, 2001, 29 p.

(IETF, 2001a) Internet Engineering Task Force (IETF), **US Secure Hash Algorithm 1 (SHA1)**, RFC 3174, setembro, 2001, 23 p.

(IETF, 2002) Internet Engineering Task Force (IETF), **Guidelines for Evidence Collection and Archiving**, RFC 3227, fevereiro, 2002, 10 p.

(IETF, 2009) Internet Engineering Task Force (IETF), **The Syslog Protocol**, RFC 5424, março, 2009, 38 p.

(ISO/IEC, 2001) INTERNATIONAL ORGANIZATION FOR STANDARDIZATION/ INTERNATIONAL ELECTROTECHNICAL COMISSION **ISO/IEC 9126:** parts 1-4 - Software Engineering-Product Quality. ISO/IEC, 2001

(ISO/IEC, 2003) INTERNATIONAL ORGANIZATION FOR STANDARDIZATION/ INTERNATIONAL ELECTROTECHNICAL COMISSION **ISO/IEC 15504-2: Information technology - Process assessment - Part 2: Performing an assessment**. ISO/IEC, 2003

(ISO/IEC, 2004) INTERNATIONAL ORGANIZATION FOR STANDARDIZATION/ INTERNATIONAL ELECTROTECHNICAL COMISSION **ISO/IEC 8601: Data elements and interchange formats - Information interchange - Representation of dates and times**. ISO/IEC, 2004

(ISO/IEC, 2005) INTERNATIONAL ORGANIZATION FOR STANDARDIZATION/ INTERNATIONAL ELECTROTECHNICAL COMISSION **ISO/IEC 25000: Systems and software Quality Requirements and Evaluation (SQuaRE) – Guide to SQuaRE**. ISO/IEC, 2005

(ISO/IEC, 2009) INTERNATIONAL ORGANIZATION FOR STANDARDIZATION/ INTERNATIONAL ELECTROTECHNICAL COMISSION **ISO/IEC 15408: Information Technology – Security Techniques – Evaluation criteria for IT security. Parts 1-3**. ISO/IEC, 2009

(ISO/IEC, 2011) INTERNATIONAL ORGANIZATION FOR STANDARDIZATION/ INTERNATIONAL ELECTROTECHNICAL COMISSION **ISO/IEC 25010: Systems and software Quality Requirements and Evaluation (SQuaRE) -- System and software quality model**. ISO/IEC, 2011

(ISO/IEC, 2011a) INTERNATIONAL ORGANIZATION FOR STANDARDIZATION/ INTERNATIONAL ELECTROTECHNICAL COMISSION **ISO/IEC 19794: Information technology -- Biometric data interchange formats**. ISO/IEC, 2011

(ITSEC, 1991) EUROPEAN COMMUNITY **Information Technology Security Evaluation Criteria (ITSEC)**, Commission of the European Communities, 1991

(JACOBS; POLL, 2011) JACOBS, B; POLL, E; **Biometrics and Smart Cards in Identity Management.** Innovating Government - Normative, Policy and Technological Dimensions of Modern Government, v. 20, n. 7, p. 419-438, 2011.

(JAQUITH, 2007) JAQUITH, A. **Security Metrics replacing fear, uncertainty, and doubt**, 1. Ed. Upper Saddle River, Addison-Wesley Professional, 2007

(JEZEQUEL; MEYER, 1997) JEZEQUEL, J. M.; MEYER, B.; **Design by Contract: The Lessons of Ariane** IEEE Computer, v. 30, n. 1, p. 129-130, 1997.

(KALAIMAGAL; SRINIVASAN, 2009) KALAIMAGAL, S.; SRINIVASAN, R. **The need for transforming the COTS component quality evaluation standard mirage to reality.** ACM SIGSOFT Software Engineering Notes, v. 34, n. 5, p. 1-4, 2009.

(KALAIMAGAL; SRINIVASAN, 2010) KALAIMAGAL, S.; SRINIVASAN, R. **Q'Facto 12 – An Improved Quality Model for COTS Components.** ACM SIGSOFT Software Engineering Notes, v. 35, n. 2, p. 1-4, 2010.

(KALAIMAGAL; SRINIVASAN, 2010a) KALAIMAGAL, S.; SRINIVASAN, R. **Q'Facto 10 – A Commercial Off-the-Shelf Component Quality Model Proposal.** Journal of Software Engineering, v. 4, n. 1, p. 1-15, 2010.

(KEYLENGTH, 2012) BLUEKRYPT, **Cryptographic Key Length Recommendation**, Disponível em: < http://www.keylength.com> Acesso em 29 de maio de 2012

(KENT; SOUPPAYA, 2006) KENT, K.; SOUPPAYA, M.; **Guide to Computer Security Log Management**, NIST special publication 800-92, setembro, 2006, p.72.

(KERNIGHAN, 1984) KERNIGHAN, B. **The Unix system and software reusability.** IEEE Transactions on Software Engineering, v. 10, n. 5, p. 513-518, 1984.

(KHAN, HAN, ZHENG, 2000) KHAN, K.; HAN J.; ZHENG, Y. **Specifying Security Requirements and Assurances of Software Components.** IN: PROCEEDINGS OF THE 5TH AUSTRALIAN WORKSHOP ON REQUIREMENTS ENGINEERING, 1., 2000, Brisbane, Australia, dezembro 2000, p. 57-65

(KHAN & HAN, 2002) KHAN, K.; HAN J. **Composing security-aware software** IEEE SOFTWARE v. 19, n. 1, p. 34-41, 2002

(KHAN & HAN, 2003) KHAN, K.; HAN J. **A Security Characterisation Framework for Trustworthy Component Based Software Systems.** IN: 27TH ANNUAL INTERNATIONAL COMPUTER SOFTWARE AND APPLICATIONS CONFERENCE (COMPSAC), 1., 2003, Dallas, TX, novembro, 2003, p. 164-169

(KHAN & HAN, 2004) KHAN, K.; HAN J. **A process framework for characterizing security properties of component-based software systems.** IN: PROCEEDINGS OF THE 2004 AUSTRALIAN SOFTWARE ENGINEERING CONFERENCE (ASWEC), 1., 2004, Melbourne, Austrália, abril, 2004, p. 358-367

(KHAN & HAN, 2005) KHAN, K.; HAN J. **Deriving Systems Level Security Properties of Component Based Composite Systems.** IN: PROCEEDINGS OF THE 2005 AUSTRALIAN SOFTWARE ENGINEERING CONFERENCE (ASWEC), 1. 2005, Brisbane, Austrália, março, 2005, p. 334-343

(KHAN & HAN, 2006) KHAN, K.; HAN J. **Assessing Security Properties of Software Components A Software Engineer's Perspective**, IN: PROCEEDINGS OF THE 2006 AUSTRALIAN SOFTWARE ENGINEERING CONFERENCE (ASWEC), 1. 2006, Sydney, Austrália, abril, 2006, p. 199-210

(KHAN & HAN, 2008) KHAN, K.; HAN J. **Specifying Security Goals of Component Based Systems - An End-User.** IN: Seventh International Conference on

Composition-Based Software Systems (ICCBSS), 1., 2008, Madrid, Espanha, fevereiro, 2008, p. 101-109

(KITCHENHAM ET AL., 2004) KITCHENHAM, B.A.; TRAVASSOS, G.H.; MAYRHAUSER, A.; NIESSINK, F.; SCHNEIDEWIND, N.F.; SINGER, J.; TAKADA, S.; VEHVILAINEN, R.; YANG, H. **Towards an Ontology of Software Maintenance**. Journal of Software Maintenance: Research and Practice, p. 365–389, 1999

(KOBLITZ, 1987) KOBLITZ, N.; **Elliptic curve cryptosystems.** Mathematics of Computation, v. 48, n. 177, p.203–209, janeiro, 1987

(KRUEGER, 1992) KRUEGER, C. **Software Reuse**. ACM Computing Surveys, v. 24, n. 2, p. 131-183, junho 1992.

(LAMPORT, 1979) LAMPORT, L.; **Constructing digital signatures from a one-way function**. Technical Report CSL-98, SRI International, outubro, 1979.

(LENZINI; TOKMAKOFF; MUSKENS, 2007) LENZINI, G.; TOKMAKOFF, A.; MUSKENS, J. **Managing Trustworthiness in Component-based Embedded Systems.** Electronic Notes in Theoretical Computer Science, v. 179, p. 143-155, julho 2007

(LI ET AL., 2009) LI, J.; CONRADI, R.; BUNSE, C.; TORCHIANO, M.; PETTER, O.; SLYNGSTAD, N.; MORISIO, M. **Development with Off-the-Shelf Components 10 Facts**. IEEE Software, v. 26, n. 2, p. 80-87, março, 2009.

(LIM, 1994) LIM, W. C. **Effect of Reuse on Quality, Productivity, and Economics.** IEEE Software, v. 11, n. 5, p. 23-30, 1994.

(LIONS, 2011) LIONS, J.L. **Ariane5 Flight Failure 501 Report by the inquiry Board** Disponível em: http://www.di.unito.it/~damiani/ariane5rep.html Acesso em: 13 de abril de 2010.

(MCCALL; RICHARDS; WALTERS, 1977) MCCALL, J. A.; RICHARDS, P. K.; WALTERS, G. F. **Factors in Software Quality**, Griffiths Air Force Base, N.Y. Rome Air Development Center Air Force Systems Command, 1977.

(MCILROY,1968) McILROY, M. D. **Mass Produced Software Components**. IN: NATO SOFTWARE ENGINEETING CONFERENCE, 1968, Garmisch, Alemanha. Anais... Brussels: Scientific Affairs Division, NATO, 1969, p. 79-85

(MENEZES ET AL., 1996) MENEZES, A. J.; OORSCHOT, P. C. V.; VANSTONE, S. A.; **Handbook of Applied Cryptography**. 1. ed. CRC Press, 1996. 780 p.

(MEYER, 2003) MEYER, B. **The Great Challenge of Trusted Components**. IN: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING (ICSE), 25., 2003, Portland, OR.

(MICROSOFT COM, 2011) Microsoft COM Technologies, Disponível em <http://www.microsoft.com/com/> Acesso em 13 de Fevereiro de 2011.

(MSFR, 1992) National Institute of Standards and Technology (NIST) **Minimum Security Functionality Requirements For Multi-User Operating Systems – MSFR**. NIST, 1992

(NETLAND ET AL., 2006) NETLAND, L. H.; ESPELID, Y.; MUGHAL, K. A.; **Security Pattern for Input Validation**. IN: 5TH NORDIC CONFERENCE ON PATTERN LANGUAGES OF PROGRAMS (VikingPLoP), 1., 2006, Helsingør, Denmark, Anais…, 2006, p. 71-79

(NEUHAUS ET AL., 2007) NEUHAUS, S.; ZIMMERMANN, T.; HOLLER, C.; ZELLER, A. **Predicting Vulnerable Software Components.** IN: 14TH ACM CONFERENCE ON COMPUTER AND COMMUNICATIONS SECURITY (CCS), 1., 2007, Alexandria, VA, **Anais…**, 2007, p. 529-540

(NIST, 1995) NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY (NIST) **An Introduction to Computer Security: The NIST Handbook**, Special Publication 800-12, 1ed, 1995, 290 p.

(NIST, 2002) NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY (NIST) **The Economic Impacts of Inadequate Infrastructure for Software Testing**, 1ed, 2002, 309 p.

(NIST, 2002a) NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY (NIST) **Secure Hash Standard**, 1ed, 2002, 75 p.

(NIST, 2003) National Institute of Standards and Technology (NIST) **Security Metrics Guide for Information Technology Systems**, NIST Special Publication 800-55, 1ed, 2003, 99 p.

(NIST, 2012) National Institute of Standards and Technology (NIST) **Recommendation for Key Management – Part 1: General (Revision 3)**, NIST Special Publication 800-55, 1ed, 2012, 147 p.

(NIST, 2009) National Institute of Standards and Technology (NIST) **Directions in Security Metrics Research**, NISTIR 7564, 1ed, 2009, 26 p.

(OCTAVE, 2012) **Operationally Critical Threat, Asset, and Vulnerability Evaluation**. Disponível em: <http://www.cert.org/octave/> Acesso 14 de abril de 2012.

(O'GORMAN, 2003) O'GORMAN, L.; **Comparing Passwords, Tokens, and Biometrics for User Authentication**. Proceedings of the IEEE, v. 91, n. 12, p. 2019 – 2040, dezembro, 2003.

(OLADIMEJI ET AL., 2006) OLADIMEJI, E. A.; SUPAKKUL, S.; CHUNG, L.; **Security Threat Modeling And Analysis: A Goal-Oriented Approach**. IN: 10th International Association of Science and Technology for Development (IASTED '06), 1., Innsbruck, Austria, 2006, **Anais…**, 2009.

(OMG, 2009) Object Management Group (OMG), Disponível em: <http://www.omg.org/> Acesso em 15 Agosto 2009.

(OMG CCM, 2002) Object Management Group (OMG), **CORBA Components**, Version 3, Document num. formal/02-06-65, Junho 2002.

(OWASP, 2012) Open Web Application Security Project (OWASP), **Data Validation**, Disponível em: <https://www.owasp.org/index.php/Data_Validation> Acesso em 10 de Janeiro de 2012.

(PERLNER, COOPER, 2009) PERLNER, R. A.; COOPER, D. A.; **Quantum Resistant Public Key Cryptography: A Survey**. IN: 8th Symposium on Identity and Trust on the Internet (IDtrust '09), 1., Gaithersburg, MD, 2009, **Anais…**, 2009, p. 85-93

(PFLEEGER, 2001) PFLEEGER, S. L. Software Engineering: Theory and Practice. 2. ed. Upper Saddle River, NJ: Prentice Hall PTR, 2001, 657 p.

(PRESSMAN, 2001) PRESSMAN, R. **Software Engineering – A Practitioner's Approach**. 5. ed. New York: McGraw-Hill Higher Education, 2001. 4, 5 p.

(PRIETO-DIAZ; FREEMAN, 1987) Prieto-Diaz, R.; Freeman, P. **Classifying Software for Reusability**. IEEE Software, v. 4, n. 1, p. 06-16, janeiro 1987

(PRIETO-DIAZ, 1993) PRIETO-DIAZ, R. **Status Report: Software Reusability**. IEEE Software, v. 10, n. 3, p. 61-66, maio 1993.

(PUTTE; KEUNING; 2001) PUTTE, T. V. D.; KEUNING, J.; **Biometrical fingerprint recognition: Don't get your fingers burned.** IN: 4th working conference on smart card research and advanced applications on Smart card research and advanced applications, 1., Bristol, IN, 2001, **Anais…**, 2001, p. 289-303.

(RABIN, 1979) RABIN, M. O.; **Digitalized signatures and public-key functions as intractable as factorization.** Technical Report MIT/LCS/TR-212, MIT Laboratory for Computer Science, janeiro, 1979.

(RIFÀ-POUS; HERRERA-JOANCOMARTÍ, 2011) RIFÀ-POUS, H.; HERRERA-JOANCOMARTÍ, J.; **Computational and Energy Costs of Cryptographic Algorithms on Handheld Devices**, Journal of Future Internet, v. 3, pp. 31-48, 2011

(RAWASHDEH; MATALKAH, 2006) RAWASHDEH, A.; MATALKAH, B. **A New Software Quality Model for Evaluating COTS Components** Journal of Computer Science, Science, v.2, pp. 373-381, 2006.

(RAVICHANDRAN; ROTHENBERGER, 2003) RAVICHANDRAN, T.; ROTHENBERGER, M. **Software Reuse Strategies and Component Markets.** Communications of the ACM, v. 46, n.8, p. 109-114, agosto 2003.

(REINEHR, 2008) Reinehr, S. S. **Reúso Sistematizado de Software e Linhas de Produto de Software no Setor Financeiro: Estudos de Caso no Brasil**. 2008. 327 p. Tese (Doutorado) - Escola Politécnica, Universidade de São Paulo, São Paulo, 2008.

(RIVEST; SHAMIR; ADLEMAN, 1978) RIVEST, R. L.; SHAMIR, A.; ADLEMAN, L. M.; **A method for obtaining digital signatures and public-key cryptosystems.** Communications of the ACM, v. 21, n. 2, p. 120–126, fevereiro, 1978.

(SAMETINGER, 1997) SAMETINGER, J. **Software Engineering with Reusable Components**. Springer Verlag, USA, 1997

(SANS, 2002) SANS Institute, **The Ins and Outs of System Logging Using Syslog**, 1ed, 2002, 53 p.

(SELBY, 2005) SELBY, R. **Enabling Reuse-Based Software Development of Large-Scale Systems**. IEEE Transactions on Software Engineering, v. 31, n. 6, p. 495-510, junho 2005.

(SHARMA; KUMAR; GROVER, 2008) SHARMA, A.; KUMAR, R.; GROVER, P. S. **Estimation of Quality for Software Components – an Empirical Approach**. ACM SIGSOFT Software Engineering Notes, v.33, i.1 pp.1-10, novembro, 2008.

(SHIN; WILLIAMS, 2008) SHIN, Y.; WILLIAMS, L. **Is Complexity Really the Enemy of Software Security?** IN: PROCEEDINGS OF THE 4TH ACM WORKSHOP ON QUALITY OF PROTECTION, 1., 2008, Alexandria, VA, **Anais…**, 2008, p. 47-50.

(SHIN; WILLIAMS, 2008a) SHIN, Y.; WILLIAMS, L. **An empirical model to predict security vulnerabilities using code complexity metrics**, IN: PROCEEDINGS OF THE SECOND ACM-IEEE INTERNATIONAL SYMPOSIUM ON EMPIRICAL SOFTWARE ENGINEERING AND MEASUREMENT, 1., 2008, Kaiserslautern, Germany, **Anais…**, 2008, p 315–317.

(SIMÃO; BELCHIOR, 2002) SIMÃO, R. P. S.; BELCHIOR, A. D. **Um Padrão de Qualidade para Componentes de Software.** In: I Simpósio Brasileiro de Qualidade de Software (SBQS), 1., 2002, Porto Alegre, RS, **Anais...**, 2002, p. 87-101

(SOFTEX, 2007) SOFTEX - SOCIEDADE BRASILEIRA PARA PROMOÇÃO DA EXPORTAÇAO DE SOFTWARE **Perspectivas de Desenvolvimento e Uso de Componentes na Indústria Brasileira de Software e Serviços**, Relatório SOFTEX-MCT-DPCT/Unicamp, março, 2007.

(SOMMERVILLE, 2001) SOMMERVILLE, I. **Software Engineering**. 6. ed. Harlow: Addison-Wesley Publishing Company, 2001. 693 p.

(STINGER, 2012) **OWASP STINGER PROJECT**, Disponível em: <https://www.owasp.org/index.php/Category:OWASP_Stinger_Project> Acesso em 10 de Janeiro de 2012.

(STRIDE, 2012) **The STRIDE Threat Model**, Disponível em: <http://msdn.microsoft.com/en-us/library/ee823878(v=cs.20).aspx> Acesso em 14 de abril de 2012.

(SZYPERSKI, 2002) SZYPERSKI, C. **Component Software - Beyond Object-Oriented Programming**. 2. ed. Londres: Addision-Wesley Publishing Company, 2002, 589 p.

(THALHEIM ET AL., 2002) THALHEIM, L.; KRISSLER, J.; ZIEGLER, P. M.; **Body Check: Biometrics Defeated**, Disponível em: <http://www.extremetech.com/computing/51286-body-check-biometrics-defeated> Acesso em: 01 de agosto de 2011

(TRIKE, 2012) **Trike framework**, Disponível em: <http://sourceforge.net/projects/trike> Acesso em 14 de abril de 2012.

(VOAS, 1999) VOAS, J. F. **Disposable Information Systems, The Future of Software Maintenance?** Journal of Software Maintenance: Research and Practice, v. 11, n. 2, p.143-150, março 1999.

(WANG ET AL., 2005) WANG, X.; YU, H.; YIN, Y. L.; **Finding Collisions in the Full SHA-1**, IN: Advances in Cryptology (CRYPTO 2005), 1., 2005, Santa Barbara, CA, **Anais…**, 2005, p. 17-36.

(WEKA, 2011) **WEKA TOOLKIT**, Disponível em: <http://www.cs.waikato.ac.nz/ml/weka> Acesso em 23 de Abril de 2011

(WINDLEY, 2005) WINDLEY, P. J.; **Digital Identity**. 1. Ed. O'Reilly Media, 2005, 238 p.

GLOSSÁRIO

Agregação de Eventos	Agrupamento em uma única entrada de eventos similares, acrescida do número de quantidade de ocorrências. (KENT; SOUPPAYA, 2006)
Análise de um Arquivo de Log	Extração de dados de um arquivo de log para análise. (KENT; SOUPPAYA, 2006)
Arquivamento de Arquivo de Log	Conservação de um arquivo de log, realizado de acordo com a importância do software e as normas internas. (KENT; SOUPPAYA, 2006)
Biometria	Identificação de pessoas a partir de características físicas, de comportamento ou habilidades profundamente enraizadas. (JACOBS; POLL; 2011)
Característica de Qualidade	Conjunto de propriedades de um produto de software, pelas quais a qualidade pode ser descrita e avaliada. A característica pode ser refinada em múltiplos níveis de subcaracterísticas. (ISO/IEC, 2001)
Cifradores de Blocos	Algoritmos cuja característica é quebrar uma mensagem de texto em blocos de tamanho *t* sobre um alfabeto *A* e realizar a cifragem de um bloco por vez (MENEZES ET AL., 1996)
Cifradores de Fluxo	Algoritmos que utilizam um fluxo de dígitos para combinar com os blocos de tamanho 1 de uma mensagem de texto realizando a cifragem de um bloco por vez (MENEZES ET AL., 1996)
Compensation (Tratamento de Erros)	Consiste em quando o estado errôneo contem redundância suficiente para mascarar o erro, não mostrando o mesmo para o usuário. (AVIZIENIS ET AL., 2004)
Componente de Software	Parte de um sistema de software quase independente e normalmente substituível, cuja responsabilidade é cumprir uma clara função dentro de uma arquitetura bem definida, através de interfaces e com dependências explícitas de contexto, podendo em tempo de execução ser dinamicamente vinculado a um ou mais programas. (BROWN, WALLNAU, 1998)
Compressão de Arquivo de Log	Compressão de arquivos de log antigos evitando a necessidade de mais espaço de armazenamento. (KENT; SOUPPAYA, 2006)
Confiabilidade	Habilidade de manter um determinado nível de desempenho ao longo do tempo quando usado em determinadas condições. Como não há um envelhecimento do software, os defeitos que ocorrem durante o processo de desenvolvimento, podem resultar em falhas no software no momento de seu uso. (ISO/IEC, 2001)
Conversão de Arquivo de Log	Transformação do formato de um arquivo de log para outro formato para fins de portabilidade. (KENT; SOUPPAYA, 2006)

Correlação de Eventos	Busca por relações entre duas ou mais entradas de um arquivo de log. (KENT; SOUPPAYA, 2006)
Criação e Checagem da Integridade do Arquivo de Log	Criação de um valor hash para cada arquivo de log armazenado. Esse valor é armazenado em um ambiente seguro para a verificação de mudanças, detectando assim possíveis manipulações do arquivo. (KENT; SOUPPAYA, 2006)
Criptografia de Dados	Estudo de técnicas matemáticas relacionadas com o aspectos de segurança da informação como confidencialidade, integridade, autenticação de entidades e autenticação de dados de origem. (MENEZES ET AL., 1996)
Deadlock	O fracasso da sincronização entre dois ou mais processos, de forma que estes não são capazes de chegar a um acordo sobre um evento comum, embora estejam dispostos a participar em outros eventos; esse conflito de sincronização ocorre durante a execução do software. (HILDERING, 2003)
Desenvolvimento Baseado em Componentes	Os passos técnicos de design e implementação necessários para desenvolver e montar componentes na forma de sistemas e entregar os sistemas resultantes. (BASS ET AL., 2001)
Diagnóstico (Tratamento de Faltas)	Localizar e identificar os itens que estão causando as faltas. (AVIZIENIS ET AL., 2004)
Dispositivos Biométricos	Forma de autenticação em sistemas a partir de características das pessoas, pois existem diferentes tipos de informação e precisão, substituindo assim outras formas de acesso (WINDLEY, 2005)
Eficiência	Habilidade de prover um desempenho apropriado, relativa ao total de recursos e tempo utilizados, sob determinadas condições. (ISO/IEC, 2001)
Engenharia de Software Baseada em Componentes	São as práticas necessárias para executar o desenvolvimento baseado em componentes de forma repetível para construir sistemas com propriedades previsíveis. (BASS ET AL., 2001)
Expressões Regulares	Textos especiais para descrever um determinado padrão, que servem para encontrar dados em uma determinada String ou realizar a validação de dados (OWASP, 2012)
Falso Negativo	Falso negativo é a notificação de não cadastrado a uma pessoa que está presente na base de dados, ou seja, com a característica cadastrada. (JACOBS; POLL; 2011)
Falso Positivo	Notificação de cadastrado a uma pessoa que não está presente na base de dados, ou seja, com a característica não cadastrada. (JACOBS; POLL; 2011)
Filtragem de Eventos de um Arquivo de Log	Remoção de eventos contidos em um arquivo de log que não sejam úteis para análise. (KENT; SOUPPAYA, 2006)

Funcionalidade	Habilidade do software de fornecer funções para atender as necessidades conhecidas e/ou implícitas quando o software é utilizado em determinadas condições. (ISO/IEC, 2001)
Gerenciamento da Chave Criptográfica	Conjunto de técnicas e processos para dar suporte ao estabelecimento e manutenção do relacionamento das chaves entre partes autorizadas (MENEZES ET AL., 1996)
Identificação de Entidade	Processo onde uma parte é assegurada, via aquisição de alguma evidência corroborativa, da identidade de uma segunda parte envolvida no protocolo, e que a segunda parte está atualmente participando, isto é, é ativa no momento em que a evidência foi adquirida (MENEZES ET AL., 1996).
Isolamento (Tratamento de Faltas)	Processo de exclusão lógica ou física dos componentes com faltas evitando sua participação no sistema, fazendo com que a falta permaneça dormente. (AVIZIENIS ET AL., 2004)
Limite de Confiança	Define que todos os softwares possuem uma camada em que se acredita que os dados estão bem formados e seguros, por validações anteriores. Assim não existe razão para que esses dados sejam novamente verificados. (HOWARD; LEBLANC, 2002)
Limpeza de Dados (Validação de Dados)	Processo de modificação da entrada do usuário em um formato aceito. Essa modificação pode ser realizada utilizando listas brancas ou listas negras (OWASP, 2012)
Listas Brancas (Validação de Dados)	Conjunto de restrições para valores bons conhecidos. A principal característica é rejeitar todos os dados que não estejam de acordo com esse conjunto de restrições. (OWASP, 2012)
Listas Negras (Validação de Dados)	Conjunto de restrições para valores ruins conhecidos. A principal característica é rejeitar todos os dados que estejam de acordo com esse conjunto de restrições. (OWASP, 2012)
Log	Forma de gravar eventos que ocorrem nos sistemas e redes de uma organização. (KENT; SOUPPAYA, 2006)
Manipulação de Dados de Entrada	Processo de validação de dados fornecido por outras entidades em um conjunto de regras pré-definidas. (NETLAND ET AL., 2006)
Manutenibilidade	Habilidade do software de ser modificado, o que inclui correções, melhoramentos ou adaptações para mudanças no ambiente, requisitos e especificações funcionais. (ISO/IEC, 2001)
Métricas	Ferramentas criadas para facilitar a tomada de decisão e melhorando a responsabilidade e o desempenho, através da coleta, análise e comunicação de desempenho de um grupo relevante de dados relacionados. (NIST, 2003)

Modelagem de Ameaças	Processo formal para identificar, documentar e mitigar ameaças de segurança a um sistema de software. Além disso, ajuda a times de desenvolvedores a entender a forma de trabalho de ameaças, observando o sistema a partir dos olhos de um potencial adversário. (OLADIMEJI ET AL., 2006)
Normalização de Arquivo de Log	Categorização dos eventos de um arquivo de log para uma representação particular auxiliando a operação de Relatório de Log. (KENT; SOUPPAYA, 2006)
Personal Identification Numbers (PINs)	Senha de quatro a oito dígitos e está associado a um token (O' GORMAN, 2003).
Política de Segurança	Conjunto de motivações e restrições para segurança, que devem ser respeitados. (NIST, 1995)
Pontos de Estrangulamento	São pontos de entrada que controlam a entrada de dados dentro do limite de confiança do sistema. Não existe uma restrição para a quantidade de pontos, contanto que todos validem os dados de entrada, evitando problemas. (HOWARD; LEBLANC, 2002)
Portabilidade	Habilidade de utilizar um software em diversas plataformas com pouco esforço de adaptação. (ISO/IEC, 2001)
Qualidade de Software	O grau que um sistema, componente de sistema ou processo, atende a determinados requisitos ou a necessidade ou a expectativa do cliente ou usuário. (IEEE, 1990).
Reconfiguração (Tratamento de Faltas)	Troca de componentes defeituosos por componentes sem faltas ativas. (AVIZIENIS ET AL., 2004)
Redução de Arquivo de Log	Remoção de entradas desnecessárias dentro de um arquivo de log, gerando um novo arquivo menor. (KENT; SOUPPAYA, 2006)
Reinicialização (Tratamento de Faltas)	Modificação, atualização e registro uma nova configuração sobre as tabelas e registros de atualizações. (AVIZIENIS ET AL., 2004)
Relatório de Arquivo de Log	Apresentação das análises realizadas, constituindo dos resumos das atividades significativas de um sistema sobre um período de tempo. (KENT; SOUPPAYA, 2006)
Reúso de Software	Processo de criação de software utilizando artefatos de software já existentes ao invés de realizar a sua construção a partir do zero. (KRUEGER, 1992)
Reúso Horizontal	Reúso de partes genéricas em diferentes domínios ou áreas de aplicação, contribuindo para o reúso entre diferentes tipos de aplicações. (PRIETO-DIAZ, 1993)
Reúso por Composição	Reúso utilizando componentes, mas sem modificá-los, construindo assim componentes de um maior nível a partir da

	combinação de componentes de menor nível. (SAMETINGER, 1997)
Reúso por Geração	A concepção de componentes para tratar as necessidades especificas de um domínio de aplicação e depois serem utilizadas em outros softwares com domínios semelhantes. (PFLEEGER, 2001)
Reúso Vertical	Reúso realizado dentro de um mesmo domínio ou área de aplicação, cujo objetivo é gerar novos modelos para uso na criação de sistemas sob uma mesma área de aplicação. (PRIETO-DIAZ, 1993)
Rollback (Tratamento de Erros)	Retorno do sistema a um estado salvo anteriormente, chamado de *chechpoint*, em que o erro ainda não estava ocorrendo. (AVIZIENIS ET AL., 2004)
Rollforward (Tratamento de Erros)	Criação de um novo estado para o sistema sem erros. (AVIZIENIS ET AL., 2004)
Rotação de Arquivo de Log	Encerramento de um arquivo de log quando é considerado completo, criando um novo arquivo para armazenar os eventos. (KENT; SOUPPAYA, 2006)
Segurança	Composição de atributos de confidencialidade, integridade e disponibilidade, exigindo a existência concorrente de 1) disponibilidade somente para ações autorizadas, 2) confidencialidade, 3) integridade com o “impróprio” significando “Não Autorizado”. (AVIZIENIS ET AL., 2004)
Token	Token é um item físico que contém informações de um determinado artefato, normalmente associado a categoria de algo de posse. (O’ GORMAN, 2003)
Usabilidade	Habilidade para o entendimento, aprendizagem e uso de um software pelo usuário, sendo este atrativo em determinadas condições. (ISO/IEC, 2001)
Visibilidade de Conteúdo – Caixa Branca	Reúso de artefatos com o poder de realizar a sua adaptação e modificação (PRIETO-DIAZ, 1993)
Visibilidade de Conteúdo – Caixa Cinza	Reúso de artefatos que revelam apenas uma pequena parte de sua implementação (SZYPERSKI, 2002)
Visibilidade de Conteúdo – Caixa de Vidro	Reúso de artefatos com a necessidade de “olhar para dentro” do componente, sem realizar alterações em seu conteúdo, descobrindo as propriedades internas principalmente quando a descrição do componente é inadequada (EZRAN; MORISIO; TULLY, 2002).
Visibilidade de Conteúdo – Caixa Preta	Reúso de artefatos sem observar, conhecer ou modificar o seu conteúdo. (PRIETO-DIAZ, 1993) e (SAMETINGER, 1997)

Visualização de Arquivo de Log	Apresentação de eventos dentro de um arquivo de log em uma forma fácil de ser entendida, podendo prover habilidades para ordenação de entradas. (KENT; SOUPPAYA, 2006)
Vulnerabilidade	Instância de uma falta na especificação, desenvolvimento ou configuração de um software, tal que sua execução pode violar uma política de segurança implícita ou explícita do sistema (SHIN; WILLIAMS, 2008)

APÊNDICE A – Tópicos de Criptografia de Dados

Algoritmos Criptográficos de Chave Simétrica

A chave simétrica é uma tupla de chaves {*e, d*} utilizadas para cifrar/decifrar uma mensagem comum (GOLLMANN, 2006). Quando é realizada a cifragem com a chave *e*, é necessário enviá-la para o destinatário, por um canal seguro, a fim de produzir a chave *d* para decifrar a mensagem, esta enviada por um canal não seguro, que pode sofrer interceptações de um adversário. O problema principal disso é o envio da chave, pois esta tem que se manter secreta para que a mensagem cifrada seja mantida em sigilo.

A Figura 6-1 mostra esse processo. Alice possui um texto sigiloso ***m*** que deseja enviar a Bob. Ela cifra o texto utilizando uma chave ***e*** produzindo uma mensagem cifrada ***c***. Ela envia a chave ***e*** por um canal seguro e a mensagem ***c*** por um canal não seguro. Bob ao receber a chave ***e*** produz uma chave ***d***. Ele utiliza a chave ***d*** para decifrar a mensagem ***c***, gerando o texto sigiloso ***m***.

Do processo há a dedução de que ambas as partes conhecem o conjunto de cifragem/decifragem. Assim como a chave ***e*** deve ser mantida em segredo, pois nos algoritmos simétricos, pode ser obtida a chave ***d*** a partir da chave ***e***. Existem duas classes de algoritmos simétricos, os *cifradores de blocos* e os *cifradores de fluxos*.

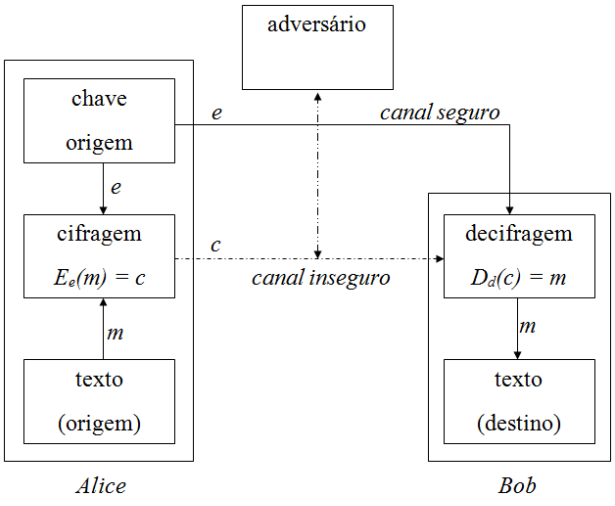

Figura 6-1. Comunicação de um algoritmo com chave simétrica, adaptado de (MENEZES ET AL., 1996).

Cifradores de blocos são algoritmos cuja característica é quebrar uma mensagem de texto em blocos de tamanho ***t*** sobre um alfabeto ***A*** e realizar a cifragem de um bloco por vez (MENEZES ET AL., 1996). São os algoritmos simétricos mais utilizados e possuem duas importantes subclasses, os *Cifradores de*

Substituição e os *Cifradores de Transposição* (MENEZES ET AL., 1996). Os primeiros são algoritmos que substituem um símbolo ou grupo de símbolos por outro símbolo ou grupo de símbolos. Os Cifradores de Transposição são algoritmos que realizam a troca de posição de símbolos de acordo com um sistema regular dentro de um bloco de tamanho ***t***, mas preserva os símbolos e é facilmente criptoanalisado, ou seja, decifrado por alguma análise feita sobre o texto criptografado.

Cifradores de fluxo são algoritmos que utilizam um fluxo de dígitos para combinar com os blocos de tamanho **1** de uma mensagem de texto realizando a cifragem de um bloco por vez (MENEZES ET AL., 1996). Esse fluxo pode ser gerado de forma aleatória ou por um algoritmo que utiliza uma semente, um pequeno fluxo de dados, sendo chamado de algoritmo gerador de fluxos de texto. Os cifradores de fluxo possuem um problema onde se o criptoanalista conhece ou adivinha parte da mensagem pode ajudar a quebra-la. Esta foi uma das formas de quebrar o protocolo WEP (*Wired Equivalent Privacy*), pois na maioria das vezes o cifrador utilizado era do tipo fluxo (CALLAS, 2006).

Para resumir esse tópico sobre algoritmos simétricos, serão descritos abaixo as vantagens e desvantagens destes. As vantagens são:

- Os algoritmos simétricos são projetados para obter altas taxas de transferência de dados. Certas implementações diretamente no hardware podem atingir taxas de centenas de megabytes por segundo enquanto que em implementações em software atingem alguns megabytes por segundo.
- As chaves criptográficas deste tipo de algoritmo são relativamente curtas
- Podem ser utilizados para construir vários mecanismos de criptografia como geradores de números e funções hash.
- Podem ser utilizados em conjunto para gerar chaves mais fortes.

As desvantagens dos algoritmos simétricos são:

- Em uma comunicação de dois computadores, a chave deve permanecer secreta nos dois lados.
- Em uma comunicação de dois computadores, a chave deve ser mudada constantemente, se possível entre cada sessão de comunicação.

Algoritmos Criptográficos de Chave Assimétrica

A criptografia assimétrica é composta por duas chaves, uma para a cifragem e outra para a decifragem (GOLLMANN, 2006). A chave pública pode ser mantida exposta ao público e a chave privada não deve ser mantida exposta ao público. Obviamente estas são algoritmicamente relacionadas, mas não deve ser possível obter a chave privada a partir de sua chave pública.

A Figura 6-2 mostra o processo utilizando um algoritmo com chave assimétrica. Alice possui um texto sigiloso ***m*** que deseja enviar a Bob. Ela avisa Bob e este seleciona o seu par de chaves criptográficas (***e***, ***d***). Bob envia a chave ***e*** (chave pública) para Alice por um canal não seguro, mas mantém a chave ***d*** (chave privada) segura. Alice cifra o texto utilizando uma chave ***e***, produzindo uma mensagem cifrada ***c***. Ela envia a mensagem ***c*** por um canal não seguro. Bob ao receber a mensagem **c** utiliza a sua chave *d* para decifrar, gerando o texto ***m***.

De acordo com Callas (2006) e Perlner e Cooper (2009) os algoritmos de criptografia assimétricos podem ser divididos em duas famílias devido a sua primitiva matemática de funções, a família de fatoração e a família de logaritmos.

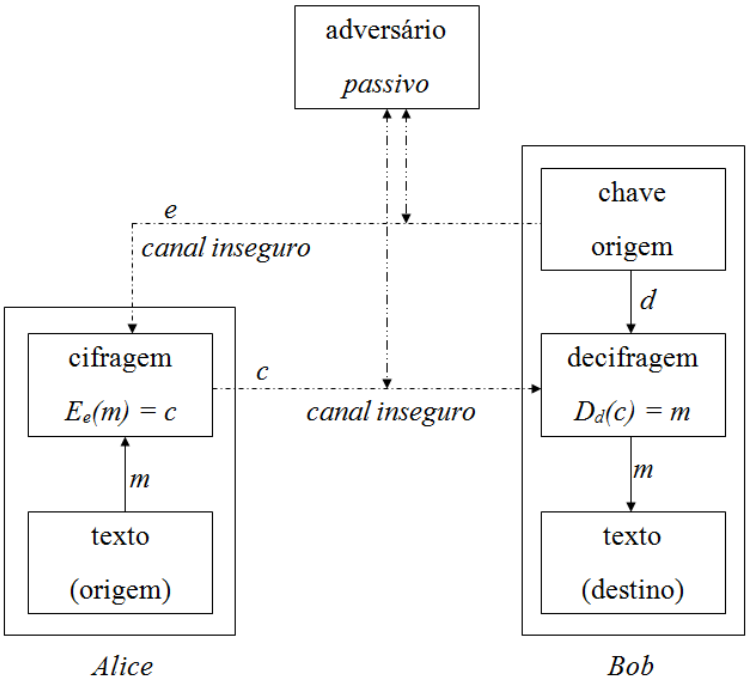

Figura 6-2. Comunicação de um algoritmo com chave assimétrica, adaptado de (MENEZES ET AL., 1996).

A família de fatoração trabalha com dois números primos de base similar, ***p*** e ***q***, que a partir da sua multiplicação obtém um valor ***n***. Esse valor é processado com formulas matemáticas sobre o texto original, obtendo o texto cifrado (CALLAS,

2006). Caso haja interceptação do texto cifrado é fácil reverter as formulas se os valores ***p*** e ***q*** forem conhecidos, mas é muito difícil se houver somente o conhecimento do valor ***n***. Somente o dono da chave privada possui o conhecimento de ***p*** e ***q***, portanto o sistema é seguro. Alguns exemplos são o RSA (RIVEST; SHAMIR; ADLEMAN, 1978) e o Rabin (RABIN, 1979).

A família de logaritmos discretos inteiros possui vários membros. O sistema básico provém do algoritmo Diffie-Hellman (1976). O seu ponto de vista é o de ser fácil calcular a fórmula básica $m = g^x$, mas é muito difícil obter o valor de ***x*** a partir dos valores ***m*** e ***g***. Esse sistema básico não era um algoritmo e sim um dispositivo conhecido como Troca de Chave. O algoritmo El-Gamal (1984) transforma esse sistema em um algoritmo criptográfico, utilizando a fórmula básica e criando pares de chaves estáticas da mesma forma que o RSA, podendo realizar a cifragem/decifragem a partir destas.

Todos os algoritmos acima se utilizam da matemática modular. Essa matemática é um sistema baseado em números inteiros que regressam ao primeiro número, zero, ao atingirem o valor do módulo (CALLAS, 2006). Operações que na maioria dos casos não geram valores inteiros, como raízes, a partir da matemática modular podem ser convertidos para um número inteiro, ao obter o resto da divisão do resultado sobre o módulo.

O algoritmo *Elliptic Curve Cryptograpy* (ECC) é uma derivação do sistema básico de *Diffie-Hellman* (KOBLITZ, 1987). Este possui dois aspectos interessantes: chaves pequenas e um rápido processamento computacional, úteis para o dia a dia. Para resumir esse tópico sobre algoritmos assimétricos, serão descritas abaixo as vantagens e desvantagens destes. As vantagens são:

- Somente a chave privada deve ser mantida em segredo.
- Dependendo da utilização, o par de chaves pode ficar anos sem ser mudado.
- Numa ampla rede, o número de chaves necessárias é menor se comparado com o número de chaves para algoritmos simétricos.

As desvantagens dos algoritmos assimétricos são:

- A taxa de transferência de dados para o mais popular algoritmo assimétrico é muitas vezes menor que o melhor algoritmo simétrico existente.

- As chaves criptográficas possuem um tamanho muito maior se comparadas com o tamanho das chaves dos algoritmos simétricos.
- Nenhum algoritmo assimétrico conseguiu provar que é totalmente seguro.

Gerenciamento da Chave Criptográfica

Gerenciamento da chave criptográfica é o conjunto de técnicas e processos para dar suporte ao estabelecimento e manutenção do relacionamento das chaves entre partes autorizadas (MENEZES ET AL., 1996). Fazem parte dessas técnicas a geração, distribuição e instalação de chaves, atualização, revogação e destruição de chaves, e o armazenamento, cópia, recuperação e arquivamento de chaves. Menezes ET AL. (1996) enumera que o ciclo de vida de uma chave criptográfica pública contém doze estados descritos na Figura 6-3.

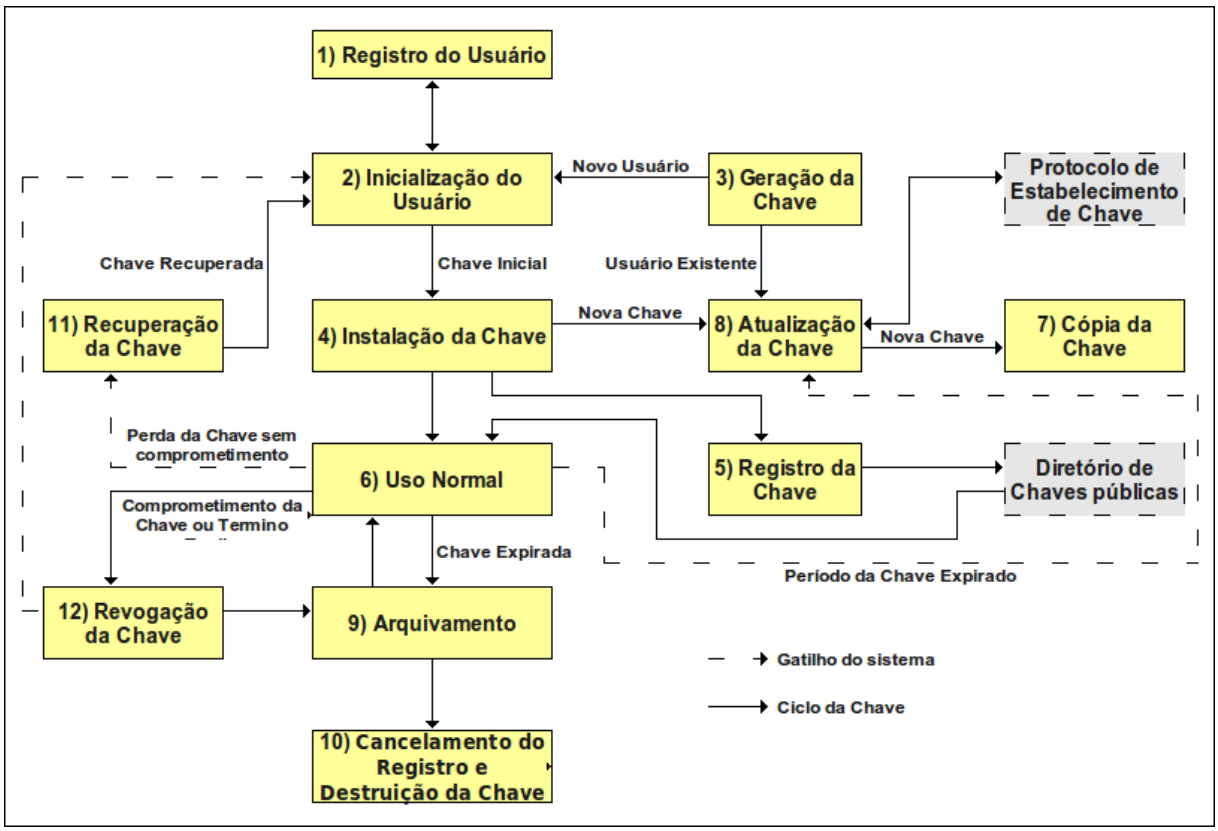

Figura 6-3. Ciclo de Vida de uma chave criptográfica pública, adaptado de (MENEZES ET AL., 1996).

1. **Registro do Usuário**: Uma entidade se torna um membro autorizado de um domínio de segurança. Isso envolve a aquisição, ou criação e troca, de um material inicial para a chave como senhas compartilhadas.
2. **Inicialização do Usuário**: A entidade inicializa sua aplicação criptográfica, envolvendo uso ou instalação do material inicial para a chave do estado 1.

3. **Geração de Chave**: Devem incluir medidas para garantir propriedades apropriadas para a intenção da aplicação ou algoritmo e aleatoriedade no sentido de ser previsível para os atacantes com probabilidade negligenciável. Uma entidade pode gerar suas próprias chaves, ou adquirir de um sistema confiável.
4. **Instalação da Chave**: O material da chave é instalado para o uso operacional dentro de um software ou hardware da entidade. Este cria uma sessão segura on-line através da qual as chaves de trabalho são estabelecidas. Durante as atualizações seguintes, novo material é instalado substituindo o em uso, através de uma atualização segura.
5. **Registro da Chave**: Em associação com a chave de instalação, o material da chave será oficialmente gravado, por uma entidade registradora, e será associado a um nome que identifica unicamente a entidade. Para chaves públicas, certificados serão criados por uma autoridade certificadora, que serve como avalista desta associação, e então é disponibilizado para outros através de um diretório público ou outros meios.
6. **Uso Normal**: O objetivo do ciclo de vida é facilitar a operação do material da chave para propostas criptográficas padronizadas. Em circunstancias normais, este estado continua até a expiração da chave, podendo ser subdividido em um ponto que a chave pública expirou, mas não a privada.
7. **Cópia da Chave**: A cópia do material da chave em um servidor de armazenamento de dados cria um ponto de origem para a recuperação da chave.
8. **Atualização da Chave**: Após a expiração da chave, o material operacional desta é substituído por um novo. Isso envolve diversos aspectos como geração, derivação com uma parte confiável. Para chaves públicas, a atualização e registro das novas chaves tipicamente envolve comunicações seguras com autoridades certificadoras.
9. **Arquivamento**: O material que não está mais em uso pode ser arquivado para prover uma recuperação da chave em circunstancias especiais.
10. **Cancelamento do Registro e Destruição da Chave**: Uma vez que não há mais requisitos para o valor da chave ou manter a associação com a entidade, esta é cancelada e todas as copias são destruídas. No caso de chaves privadas, todos os itens são apagados seguramente.

11. **Recuperação da Chave**: Se o material é perdido de forma que não comprometa a segurança, é possível restaurar por uma cópia segura.
12. **Revogação da Chave**: Pode ser necessário remover as chaves do uso operacional, antes do seu termino originalmente programado, por comprometimento da chave. Para chaves públicas distribuídas por certificados isso envolve a revogação dos certificados.

Outro ponto a ser destacado é a forma de armazenamento da chave criptográfica, pois sem uma segurança adequada sobre o sistema de armazenamento, pode causar a obtenção das chaves ali armazenadas. Existem diversas formas de armazenamento das chaves, o estudo feito por Chen e Zhou (2010) retrata alguma dessas formas e faz comparações de tempo e formas de segurança. É interessante remeter a esse tema, pois a segurança pode variar conforme o tipo da chave a ser armazenada. O autor apresenta nesse estudo algumas considerações sobre chaves simétricas e assimétricas, assim como sua distribuição, autenticação autorização e integridade dos dados criptografados.

Algoritmos Hash

Funções *Hash* ou *Message Digests* formam uma parte importante da criptografia com objetivo específico de integridade de dados e autenticação de mensagens (MENEZES ET AL., 1996). Uma função *hash* tem como entrada uma string de tamanho variável e como saída, chamada de *código-hash* ou *valor-hash*, uma String, normalmente número, de tamanho fixo. Assim a idéia básica que a função *hash* produz uma imagem representativa, também chamada de impressão digital, de uma entrada, relacionando esta com o *código-hash* criado.

Como há tamanho limitado de caracteres para a geração do *código-hash* é inevitável que ocorram colisões, ou seja, que dois strings de entrada diferentes produzam a mesma saída. Para uma função *hash* ser considerada útil para a criptografia ela deve possuir algumas propriedades (CALLAS, 2006):

- Deve ser difícil reverter uma função *hash*, ou seja, tendo o número *hash* deve ser difícil achar a string de entrada que o gerou.
- Tendo um número *hash* deve ser difícil identificar uma string geradora, sendo que a relação entre o número e a string deve ser a mais opaca possível.

- Tendo uma string geradora, deve ser difícil obter outra string geradora em que seus números *hash* colidam.

A típica utilização de uma função hash é a verificação de intergridade de um determinado dado. Esta ocorre da seguinte maneira: Um dado ***x*** foi submetido a uma função *hash*, gerando um *código-hash* ***c***. Este foi armazenado em um local seguro indicando que pertence a ***x***. Passado um tempo ***t***, ***x*** é submetido à mesma função, gerando um código ***c1***. Caso ***c*** seja diferente de ***c1***, implica que ***x*** foi alterado, abrindo margem para uma investigação sobre violações no acesso a ***x***.

No nível mais alto as funções *hash* são divididas em duas classes: *Funções hash convencionais* e *funções hash com chave.* A primeira recebe como parâmetro apenas a mensagem a ser gerado o *código-hash*. A segunda classe recebe como parâmetro a mensagem e uma chave secreta.

Dentro da classe de funções *hash* convencionais, existe um tipo chamado *Modification Detection Codes* (MDCs). Esse é dividido em outros dois tipos, as funções *one-way hash functions* (OWHFs), onde achar uma mensagem de entrada que possui um determinado *código-hash* é muito difícil, e as funções *collision resistant hash functions* (CRHFs), onde achar duas mensagens de entrada que possuam o mesmo *código-hash* é muito difícil.

Alguns exemplos de algoritmos *hash* são o MD5 (IETF, 1992) com geração de *código-hash* de 128 bits, SHA1 (IETF, 2001a) com geração de *código-hash* de 160 bits, criado pelo NIST para substituir o MD5 e o RIPEMD-160 (DOBBERTIN ET AL., 1996) com geração de *código-hash* de 160 bits, criado pelo RIPE (*Réseaux IP Européens*) para substituir o MD5 e por fim os algoritmos da família “SHA2” (NIST, 2002a): SHA-256, SHA 384 e SHA512 todos criados pelo NIST.

Os problemas principais das funções *hash* são as colisões, pois impactam sobre o *código-hash* fazendo-o perder a unicidade. Essas podem ser causadas pelo birthday attack, que é o ataque baseado na hipótese “quantas pessoas são necessárias dentro de uma sala para que pelo menos duas pessoas façam aniversário no mesmo dia?”. Um estudo deste ataque é apresentado em Wang et Al. (2005).

s

APÊNDICE B – Tópicos em Identificação e Autenticidade

Senhas - Passwords

Processos convencionais que utilizam senhas que não variam com o tempo têm por definição uma fraca autenticação (MENEZES ET AL., 1996). A ideia básica de uma senha está sobre a forma de um conjunto de caracteres que o usuário guarda na memória. Assim para o acesso o usuário insere um par de (usuário, senha), onde o usuário é a identidade e a senha é a evidência. O sistema verifica se as informações estão de acordo e deixa o usuário acessar o sistema.

Existem algumas técnicas para a utilização de senhas dentro de um sistema. Menezes et al. (1996) apresenta algumas delas abaixo:

- **Armazenamento de Arquivos de Senhas**: Forma mais óbvia de armazenar é a criação de um arquivo de senhas não criptografado. O sistema utiliza o arquivo para comparar usuário e senha. Uma grande desvantagem é não prover proteção contra usuários administradores.
- **Armazenamento de Arquivo de Senhas Criptografado**: É realizada uma criptografia de uma via, ou seja, que não pode ser revertida, na senha e armazena no arquivo de senhas. Assim para comparar o sistema deve obter a senha informada, criptografá-la e comparar com a senha no arquivo. Não há proteção contra usuários administradores.
- **Regras de Senha**: É mais um controle do que uma técnica para a não criação de senhas fracas. Algumas regras são comuns como tamanho da senha, variação de formatos de caracteres, verificação da senha em dicionários de ataques e data de validade, obrigando o usuário a mudá-la de tempos em tempos.
- **Retardar o processamento após algumas tentativas**: Essa técnica consiste em retardar o processamento de comparação de senhas após algumas tentativas do usuário. O objetivo é evitar ataques de força bruta.
- **Passphrases**: Ao invés de criar senhas curtas de palavras abreviadas porque não criar uma frase longa? Esse é o objetivo desta técnica para melhorar a segurança, pois é muito mais difícil quebrar uma senha de 40 caracteres do que uma de oito caracteres. Existe problemas no armazenamento da senha e no tempo para digitá-la.

PINs - Personal Identification Numbers

Os *Personal Identification Numbers* (*PINs*) não variam com o tempo. *PIN* é uma senha de quatro a oito dígitos e está associado a um *token* (O' GORMAN, 2003). Isso é chamado autenticação multi-fator. *Token* é um item da categoria "Algo de Posse". Exemplos são os cartões magnéticos ou com chip. Para garantir o acesso deve ser inserido o PIN correto ao utilizar o *token*. Assim possui dois níveis de segurança, sendo necessário ter o *token* e o PIN para utilizá-lo.

Como é uma senha muito pequena, invasores podem quebrá-la através de exaustão, ou seja, vão tentando senha por senha até descobrir. Para evitar isso, as administradoras de cartão inserem um mecanismo de segurança para caso for inserida três senhas erradas em sequência o cartão é bloqueado ou desativado.

Cartões com chip também são chamados de *smart cards*, pois possuem uma interface simples para acessar a memória ou o processador do chip. O cenário mais básico para o uso de *smart cards* é o armazenamento de um certificado digital, melhorando a portabilidade. Assim o cartão pode utilizar sua memória e processador internos para cifrar e decifrar tarefas, através do acesso à sua chave privada, não realizando o trânsito das chaves criptográficas para fora do cartão, minimizando as chances de ataque.

Senhas Únicas – One-time Passwords

Uma progressão natural dos processos de senha fixa para os processo de desafio-resposta, pois as senhas fixas têm como principal problema de um atacante observar a senha e utilizá-la. Uma solução parcial está nas senhas únicas, onde cada senha pode ser usada apenas uma vez. Menezes et al. (1996) apresenta algumas formas de construir um processo de senha única.

- **Listas compartilhadas de senhas únicas**: Usuário e sistema usam um conjunto de **t** senhas secretas, cada uma válida para uma única autorização. O problema é a manutenção desse conjunto, identificando senhas já utilizadas. Uma variação envolve a criação de uma tabela de desafio-resposta, onde os pares valem apenas uma autorização.
- **Atualização sequencial de senhas únicas**: Inicialmente somente uma senha única é compartilhada. Durante o processo de autorização o usuário cria e transmite para o sistema uma nova senha, a qual é gerada a partir

de uma chave criptográfica atual. Caso ocorra uma falha na comunicação esse método fica prejudicado.

- **Senhas únicas sequenciais baseadas em uma função de uma via**: É o que o processo Lamport (1979) utiliza para gerar suas chaves criptográficas. É um método mais eficiente que o método anterior e pode ser considerado como um protocolo de desafio-resposta onde o desafio é implícito para a posição corrente com a sequência de senhas.

Identificação por Desafio-Resposta – Challenge-Response Identification

A ideia do protocolo de identificação por desafio-resposta é o que uma entidade, chamada de requerente, “prova” sua identidade para outra, chamada de verificador, ao demonstrar o conhecimento de um segredo associado com o requerente, não revelando este durante o processo. É realizado ao prover a resposta a um desafio, variando com o tempo, onde esta depende do segredo.

O desafio é tipicamente um numero escolhido por uma entidade, que é definida de forma aleatória e secretamente no inicio do protocolo. Se a linha de comunicação é monitorada por adversários, a resposta de um desafio não gera informações úteis para uma nova identificação já que esta muda na próxima. Algumas formas de gerar os desafios são apresentadas por Menezes et al. (1996):

- **Números aleatórios**: São utilizados para prover garantias de unicidade e imprevisibilidade, impedindo ataques de *chosen-text*, onde o atacante tem o objetivo de ganhar informações adicionais, reduzindo a segurança do sistema. Eles são utilizados da seguinte forma: Ao criar uma mensagem uma entidade insere um número randômico. Para a próxima mensagem é necessário um conhecimento do último número gerado, pois é necessário para gerar o próximo número.
- **Números sequenciais**: Um número sequencial ou número serial tem o objetivo de identificar com um número único uma mensagem, sendo tipicamente utilizado para detectar repetições. Números sequenciais são específicos para um par particular de entidades e devem estar associados explicitamente entre a origem e o destino da mensagem. Esta é aceita somente se o número ainda não foi utilizado satisfazendo a política de mensagens. Uma desvantagem é que cada par de entidades deve ter uma lista própria de números, torna-se lenta para grupos grandes.

- **Timestamps**: Utilizados para prover garantias de unicidade e imprevisibilidade para detectar mensagens repetidas ou criar privilégios por tempo limitado e detectar atrasos forçados. É utilizado da seguinte forma: A entidade origem obtém um *timestamp* de seu relógio local, inserindo este cifrado na mensagem. Ao receber a mensagem, a entidade destino obtém um *timestamp* de seu relógio local e subtrai do *timestamp* da mensagem. Esta deverá ser aceita se a diferença estiver em um intervalo de aceitação (por exemplo, entre 10 milissegundos a 20 segundos) ou for a primeira mensagem com timestamp da entidade origem. Uma desvantagem é a necessidade dos relógios serem seguros e sincronizados.

Os protocolos de desafio-resposta podem ser construídos com técnicas de criptografia de algoritmos simétricos ou assimétricos. Para os simétricos as duas entidades compartilham a chave, sendo que em um pequeno grupo de usuários podem ser geradas chaves para todas as combinações de pares. Existem três formatos de utilização das chaves para o protocolo de desafio-resposta (MENEZES ET AL., 1996). A primeira é o protocolo baseado na cifragem por chave simétrica, a segunda é o protocolo baseado em funções simétricas de única via e por fim o protocolo baseado em funções simétricas de única via com geradores de senha.

O protocolo de desafio-resposta utilizando algoritmos criptográficos assimétricos pode ser realizado de duas maneiras (MENEZES ET AL., 1996). O requerente decifra um desafio com a sua chave pública ou assina digitalmente um desafio.

Biometria

A definição básica de biometria é a identificação de pessoas a partir de características físicas, de comportamento ou habilidades profundamente enraizadas (JACOBS; POLL; 2011). Características físicas são o reconhecimento de face, impressão digital, geometria da mão, características da íris e da retina e o DNA. Características de comportamento são itens aprendidos ou adquiridos como a forma de escrita, assinaturas, reconhecimento de voz, forma de caminhar ou pelo ritmo de escrita em um teclado de computador.

Os dispositivos biométricos são a oportunidade de construir uma forma de autenticação em sistemas a partir de características das pessoas, pois existem diferentes tipos de informação e precisão, substituindo assim outras formas de acesso (WINDLEY, 2005). Para aplicar a biometria é necessário adotar uma característica particular, como a impressão digital, e construir um banco de dados de pessoas que utilizam o software. Esses são úteis para a área de segurança pública como na identificação de criminosos.

Existe um problema principal em todas as características biométricas que se encontra na comparação de características: os falsos positivos e os falsos negativos (JACOBS; POLL; 2011). Falso positivo é a notificação de cadastrado a uma pessoa que não está presente na base de dados, ou seja, com a característica não cadastrada. Falso negativo é a notificação de não cadastrado a uma pessoa que está presente na base de dados, ou seja, com a característica cadastrada. Para cada característica há uma taxa de ocorrência de falsos positivos e falsos negativos, mas todas são diretamente proporcionais a taxa de crescimento da base de dados.

Outro problema que existe é a falsificação de uma característica biométrica. Putte e Keuning (2001) citam a produção de uma cópia de borracha de uma impressão digital para o uso nos leitores biométricos. Thalheim et al. (2002) cita outra falha de um leitor de impressão digital no respirar sobre o leitor, e este reconhecendo o resíduo da impressão deixada pelo último usuário.

A incapacidade de inscrever certas pessoas em uma base de dados é outra questão, pois essa se encontra em anormalidades de determinadas características biométricas como quando a pessoa não possui algum dedo ou não possui a mão, impossibilitando o cadastramento, podendo excluir ou discriminar determinados grupos de indivíduos (WINDLEY, 2005). Por fim existe um problema ético, na privacidade das pessoas, ou seja, o quanto uma determinada característica revela sobre as pessoas e o quão sensível essa informação é (JACOBS; POLL; 2011).

Assinaturas Digitais

A Assinatura Digital proporciona um meio para uma entidade vincular a sua identidade a uma determinada informação. Para isso é necessário combinar uma criptografia assimétrica com funções *hash*.

Windley (2005) descreve o processo de geração de uma assinatura digital: É gerado um *código-hash* do documento a ser assinado digitalmente, esse código é

cifrado com a chave de um algoritmo assimétrico e enviado com o documento não criptografado. Chegando ao destino o número é decifrado sendo comparado com o resultado da função *hash* sobre o documento. Se forem iguais o documento não foi alterado e foi emitido pelo proprietário da chave pública.

Quando utilizado para esse propósito a chave privada é chamada de chave de assinatura e chave pública de chave de verificação. Menezes et al. (1996) cita que o objetivo principal dos ataques a assinaturas digitais é a cópia de assinaturas, ou seja, produzir assinaturas falsas que serão aceitas como se fossem de outra entidade. Existem algumas maneiras de quebrar uma assinatura:

- **Quebra total**: Um atacante possui habilidade de processar a informação da chave privada do assinante ou encontra um algoritmo eficiente com a funcionalidade de assinar equivalente ao algoritmo de assinatura válido.
- **Falsificação seletiva**: Um atacante é capaz de criar uma assinatura válida para uma mensagem particular ou classes de mensagens escolhidas a priori. Esta criação não envolve diretamente o assinante legitimo.
- **Falsificação existencial**: Um atacante é capaz de forjar uma assinatura para pelo menos uma mensagem. Ele possui pouco controle sobre a mensagem, e o assinante legitimo pode estar envolvido na fraude.

Além disso, existem dois tipos de ataques, o ataque sobre somente a chave, onde o atacante conhece somente a chave pública do assinante e o ataque a mensagem, onde o atacante possui habilidade de examinar assinaturas de mensagens conhecidas, que podem ser escolhidas pelo atacante.

Certificados Digitais

Um certificado digital agrega informações de identificação e uma chave assimétrica. Ambos são assinados por uma parte terceira, transformando em um item de autenticação (WINDLEY, 2005). A maneira mais simples de uso é através de um protocolo de desafio-resposta, substituindo o algoritmo secreto. O servidor gera uma string aleatória e envia para o usuário, este assina digitalmente a string e a retorna para o servidor. Este verifica se a string é a mesma e com isso comprova o dono do certificado. Além das vantagens da criptografia, vários certificados podem ser associados a uma única conta, armazenando diferentes tipos de atributos para cada certificado.

Existem alguns empecilhos para a utilização do certificado digital. O primeiro problema está na sua utilização, sendo um pouco mais difícil que o uso de senhas, por exemplo. O segundo problema está no custo para a produção, pois o usuário deve renovar o certificado, normalmente isso ocorre entre um a três anos dependo do nível de segurança do algoritmo, sendo muito caro para uma empresa ter o certificado como autenticação primária. Por fim, não melhora a segurança significativamente (WINDLEY, 2005).

Ataques em protocolos de identificação

Esse tópico resume as principais formas de ataque com alguma forma de prevenção (MENEZES ET AL., 1996). Essas formas são listadas abaixo com os nomes mantidos em inglês, facilitando a busca por novos materiais:

- **Impersonation**: Fraude onde uma entidade tenta se passar por outra.
- **Replay attack**: Uso de uma informação obtida a priori, de forma ilícita, sobre uma entidade verificadora. Prevenção: Uso de técnicas de desafio-resposta, números randômicos. Incorporar identidade alvo na resposta.
- **Interleaving attack**: Uso de informações obtidas a priori, de forma ilícita, sobre um ou mais protocolos de identificação, sendo sessões paralelas. Prevenção: Reunir todas as mensagens de um protocolo em execução, por exemplo, usando números randômicos encadeados.
- **Reflection attack**: *Interleaving attack* com informação de uma execução de protocolo de volta para a entidade origem. Prevenção: Incorporar identificação da entidade destino no desafio e chaves unidirecionais.
- **Forced delay**: Uma mensagem é interceptada por um atacante, tipicamente contendo um número sequencial, e este atrasa a mensagem. Esse atraso serve para realizar o roubo de informações ou a sua modificação. Note que a mensagem não é repetida. Prevenção: Combinar o uso de números randômicos com respostas de tempo curto.
- **Chosen-text attack:** Um ataque sobre um protocolo de desafio-resposta onde são escolhidos desafios para extrair informações de uma entidade requerente. Prevenção: Colocar em cada desafio-resposta um número randômico ao acaso.

APÊNDICE C – Tópicos em Auditabilidade e Log

Operações Sobre os Arquivos de Log

As operações de log consistem em processos a serem realizados pelo próprio componente ou software externo sobre os arquivos de log. Algumas das principais operações de log são definidas por (KENT; SOUPPAYA, 2006):

- **Análise de um Arquivo de Log**: Consiste na extração de dados de um arquivo de log para análise. Pode aplicar algumas operações sobre os dados extraídos como Filtragem, Agregação e Correlação de Eventos, Conversão, Normalização e Relatório de Arquivo de Log.
- **Filtragem de Eventos**: Consiste na remoção de eventos contidos em um arquivo de log que não sejam úteis para análise.
- **Agregação de Eventos**: Consiste no agrupamento em uma única entrada de eventos similares, acrescida do número de quantidade de ocorrências.
- **Correlação de Eventos**: Consiste na busca por relações entre duas ou mais entradas de um arquivo de log. A forma mais comum de aplicação é a utilização de regras para avaliar características como data, hora, endereços de IP e tipos de eventos. Pode ser automatizada por ferramentas de visualização ou métodos estatísticos. Após a conclusão é gerado um novo arquivo reunindo as informações encontradas.
- **Conversão de Arquivo de Log**: Consiste na transformação do formato de um arquivo de log para outro formato para fins de portabilidade.
- **Normalização de Arquivo de Log**: Consiste na categorização dos eventos de um arquivo de log para uma representação particular auxiliando a operação de Relatório de Log.
- **Relatório de Arquivo de Log**: Consiste na apresentação das análises realizadas. É utilizada para construir resumos das atividades significativas de um sistema sobre um período de tempo ou verificar informações relacionadas a um evento.
- **Rotação de Arquivo de Log**: Consiste no encerramento de um arquivo de log quando é considerado completo, criando um novo arquivo para armazenar os eventos. Pode ser realizada através de um agendamento (diário, semanal, mensal) ou quando o arquivo atingir um tamanho

máximo. Durante essa rotação podem ser realizadas operações de filtragem de eventos em busca de atividades maliciosas sobre o sistema.

- **Arquivamento de Arquivo de Log**: Consiste na conservação de um arquivo de log, realizado de acordo com a importância do software e as normas internas. Existem duas formas de realizar a operação. A primeira é a Retenção do Arquivo de Log onde o arquivo é tratado de forma idêntica aos arquivos de log do sistema operacional. A segunda é a Preservação do Arquivo de Log, realizando a manutenção dos arquivos que seriam descartados, por possuírem informações de interesse para investigações.
- **Compressão de Arquivo de Log**: Consiste na compressão de arquivos de log antigos evitando a necessidade de mais espaço de armazenamento. Pode ser realizada durante as operações de Rotação de Arquivo de Log ou Arquivamento de Arquivo de Log.
- **Redução de Arquivo de Log**: Consiste na remoção de entradas desnecessárias dentro de um arquivo de log, gerando um novo arquivo menor. Pode ser realizada antes da operação de Compressão de Arquivo de Log ou da operação de Análise de Arquivo de Log.
- **Criação e Checagem da Integridade do Arquivo de Log**: Consiste na criação de um valor *hash* para cada arquivo de log armazenado. Esse valor é armazenado em um ambiente seguro para a verificação de mudanças, detectando assim possíveis manipulações do arquivo.
- **Visualização de Arquivo de Log**: Consiste na apresentação de eventos dentro de um arquivo de log em uma forma fácil de ser entendida, podendo prover habilidades para ordenação de entradas.

Desafios no Gerenciamento de Arquivos de Log

Eventos de log são importantes para detectar atividades suspeitas e ataques aos softwares, mas durante esse processo existe uma captura de eventos, ações e comportamento de usuários legítimos no sistema, podendo violar sua privacidade (SANS, 2002). Assim uma política de segurança clara e bem definida deverá ser criada para explicar como é coletada, manipulada e utilizada essa informação.

Caso algum desses eventos seja utilizado para processar alguém por atividades não autorizadas, evidências devem ser geradas provando a acusação. Para isso a RFC (*Request for Comment*) 3227 (IETF, 2002) indica que uma

evidência deve conter as qualidades de ser admissível, autentica, completa, confiável e acreditável.

O guia de gerenciamento de arquivos de log descrito por Kent e Souppaya (2006) propõe alguns desafios que as corporações enfrentam para gerenciar seus arquivos. Alguns desses itens serão descritos abaixo:

- **Muitas fontes de log**: Normalmente a produção de eventos para arquivos de log estão localizados em diversos pontos dentro de uma corporação, necessitando um gerenciamento desses e assim como apenas um ponto pode gerar mais de um arquivo de log, dependendo da atividade.
- **Conteúdo inconsistente**: Cada arquivo de log contém eventos com somente um pedaço de informação, como endereços de IP ou nomes de usuários. Caso haja necessidade de uma comparação entre dois ou mais arquivos de sistemas diferentes, podem estes ter conteúdos inconsistentes como apenas um conter somente o numero do usuário sem o endereço de IP. Ou representar a mesma informação, mas de formas diferentes como transferência FTP em um arquivo e a da porta de FTP em outro.
- **Tempo de eventos díspares**: Cada ponto de log dentro de um sistema pode ter relógios internos ajustados de formas díspares, causando problemas na análise entre arquivos diferentes.
- **Formatos de Log inconsistentes**: É a não existência de um padrão para o formato de arquivos de log entre os sistemas existentes como CSV, XML ou binários, dificultando a análise entre arquivos de sistemas diferentes.

Para uma corporação gerenciar melhor seu portfólio de logs o desafio mais importante está no formato dos arquivos gerados pelos vários sistemas existentes. O próximo tópico irá descrever esses formatos.

Formato de Log

Uma das formas de melhorar os desafios do log está na adoção de um padrão de formato de log. Qualquer software produzido terá a mesma forma de saída, centralizando as operações de análise de log e verificação de ataques. Um formato criado em 1980 por Eric Allman e que ganhou muita notoriedade na última década é o s*yslog*, que foi desenhado para realizar logs de eventos de um software chamado sendmail do sistema operacional BSD (*Berkley Software Distribution*).

Pela natureza flexível do sistema BSD, o s*yslog* passou a ser utilizado em diversos softwares e dispositivos. Em agosto de 2001 o conselho IETF (*Internet Engineering Task Force*) criou a RFC 3164 (IETF, 2001) sendo o primeiro passo para prover um padrão consistente de software de log de eventos (SANS, 2002). Em março de 2009 a RFC 3164 foi substituída pela RFC 5424 (IETF, 2009).

Cada evento de log realizado pelo s*yslog* é dividido em duas partes. A primeira se chama *selector*, que é dividida em duas partes a *facility* e a *priority*. A segunda parte é chamada de *action*, formando o seguinte modelo:

[<facility>.<priority>] <action>

O *facility* serve para o s*yslog* diferenciar a origem da mensagem e quem ou o que a gerou. Existem 24 possíveis *facility* e estes não especificam o software, mas são especificações do sistema. O Quadro 6-1 mostra todos os *facilities*.

Quadro 6-1. Facilities do Syslog (IETF, 2009)

Código	**Nome**	**RFC 3164 Facility**
0	Kern	Kernel messages
1	User	User-level messages
2	Mail	Mail messages
3	Daemon	System daemons
4	Auth	Security/authorisation messages
5	Syslog	Messages generated internally by syslogd
6	Lpr	Line printer subsystem
7	News	Network news subsystem
8	UUCP	UUCP subsystem
9	Cron	Clock daemon
10	Authpriv	Security/authorisation messages
11	FTP	FTP daemon
12	-	NTP daemon
13	-	Log audit
14	-	Log alert
15	-	Clock daemon
16	Local0	Local use 0
17	Local1	Local use 1
18	Local2	Local use 2
19	Local3	Local use 3
20	Local4	Local use 4
21	Local5	Local use 5
22	Local6	Local use 6
23	Local7	Local use 7

O *priority* serve para o *syslog* indicar níveis de importância para os eventos ajudando o analista a quantificar o aparecimento de mensagens durante análise do arquivo de log. Existem oito níveis de importância descritos pelo *syslog*. O Quadro 6-2 mostra os oito *priorities.*

Quadro 6-2. Priorities do Syslog (IETF, 2009)

Código	Nome	RFC 3164 Severity
0	Emerg	Emergency: system is unusable
1	Alert	Alert: action must be taken immediately
2	Crit	Critical: critical condition
3	Err	Error: error condition
4	Warn	Warning: warning condition
5	Notice	Notice: normal but significant condition
6	Info	Informational: informational messages
7	Debug	Debug: debug-level messages

Outro formato aceito pelo *syslog* está na soma dos valores de *facility* e *priority*, podendo distinguir pelo Quadro 6-3 quais são os itens formadores.

Quadro 6-3. Combinação de Facilities e Priorities do Syslog (IETF, 2009)

		Severity Level							
Facility Level		**Emerg**	**Alert**	**Critical**	**Error**	**Warn**	**Notice**	**Info**	**Debug**
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Kernel	0	0	1	2	3	4	5	6	7
User	1	8	9	10	11	12	13	14	15
Mail	2	16	17	18	19	20	21	22	23
System	3	24	25	26	27	28	29	30	31
Security	4	32	33	34	35	36	37	38	39
Syslog	5	40	41	42	43	44	45	46	47
Line Printer	6	48	49	50	51	52	53	54	55
Network News	7	56	57	58	59	60	61	62	63
UUCP	8	64	65	66	67	68	69	70	71
Clock	9	72	73	74	75	76	77	78	79
Security	10	80	81	82	83	84	85	86	87
FTPd	11	88	89	90	91	92	93	94	95
NTPd	12	96	97	98	99	100	101	102	103
Log audit	13	104	105	106	107	108	109	110	111
Log alert	14	112	113	114	115	116	117	118	119
Clock daemon	15	120	121	122	123	124	125	126	127
Local0	16	128	129	130	131	132	133	134	135
Local1	17	136	137	138	139	140	141	142	143
Local2	18	144	145	146	147	148	149	150	151
Local3	19	152	153	154	155	156	157	158	159
Local4	20	160	161	162	163	164	165	166	167
Local5	21	168	169	170	171	172	173	174	175
Local6	22	176	177	178	179	180	181	182	183
Local7	23	184	185	186	187	188	189	190	191

A parte *action* representa a mensagem do evento, descrevendo o tempo e o texto. Com o padrão fica mais fácil para realizar a portabilidade entre sistemas, o trabalho de análise e verificação de segurança dos analistas. Existe um item a ser

descrito melhor que é a parte de segurança para o arquivo de log. Na próxima seção será descrito essa parte de segurança para o arquivo de log.

Segurança para arquivos de log

Sobre cada arquivo de log gerado informações importantes sobre o sistema, acessos de usuários, erros, tentativas de ataque são mantidas nele. Assim é necessária uma forma segura de armazenar e acessar este arquivo. Sans (2002) descreve alguns itens para que o armazenamento seja realizado com segurança:

- **Imprimir o Arquivo de Log**: Uma das melhores maneiras de manter a segurança é imprimir o arquivo de log, pois não é um meio eletrônico de armazenamento, sendo utilizado muito em sistemas de missão critica. O problema, além do gasto de papel, é a perda da habilidade de analisar com mais detalhe.
- **Dispositivo WORM**: Uma alternativa está na gravação de mídias WORM (*Write Once, Read Many*), como CD-Roms ou DVDs, pois não se pode destruir eletronicamente esses dados. Existe apenas o gasto de várias mídias de acordo com o tempo.
- **Partição Separada**: Um dispositivo de armazenamento como um HD pode ser dividido em partições, sendo utilizado para armazenar os arquivos de log. Basta ter um privilégio de administrador para montar a partição, assim poucas pessoas poderão desmontar a partição ou modificar seus dados.
- **Linha Serial**: Uma máquina anexada via linha serial na qual serão armazenados os arquivos de log é uma excelente alternativa, pois basta ter uma grande capacidade de armazenamento e acesso à maquina para realizar as análises e proteger a máquina de ataques isolados.

Número Hash: Realizar a operação de Criação e Checagem da Integridade do Arquivo de Log descrita no item de operações de arquivos de log é uma forma de melhorar a segurança sobre os arquivos.

APÊNDICE D – Tópicos em Ameaças e Vulnerabilidades

Origem de Faltas

Para caracterizar uma falta é necessário estabelecer a sua origem. Avizienis et al. (2004), apresenta uma visão de oito classes que representam pontos de vista para a geração de faltas. Essas classes são as seguintes:

1) **Fase de Criação:** Define em qual fase a falta ocorreu. Pode ser caracterizada em fase de desenvolvimento (criação/manutenção) e a fase operacional (utilização)
2) **Fronteiras do Sistema:** Define se a falta ocorreu dentro das fronteiras do sistema, sendo chamada de falta interna, ou fora das fronteiras do sistema, sendo chamada de falta externa.
3) **Causa Fenomenológica:** Define se a falta ocorreu a partir de um evento da natureza, como um raio, queda de energia, sendo chamada de Falta Natural ou se ela foi causada por um ato humano, sendo chamada de Falta por Ato Humano.
4) **Dimensão:** Define se a falta foi causada por um erro na parte de software, sendo chamada de Falta por Software ou se iniciou na parte de hardware, sendo chamada de Falta por Hardware.
5) **Objetivo:** Define se a falta foi realizada de forma a causar malefícios ao sistema, sendo chamada de Falta Maliciosa, ou se a falta não foi intencional, sendo chamada de Falta Não Maliciosa.
6) **Intenção:** Define se a falta foi iniciada por decisões erradas, sendo chamada de Falta Deliberada ou se foram ações involuntárias sem a consciência do ator, sendo chamada de Faltas Não Deliberadas.
7) **Capacidade:** Define se a falta foi inserida por acidente, sendo chamada de Falta Acidental ou se foi inserida por incompetência, sendo chamada de Falta por Incompetência.
8) **Persistência:** Define se a falta é Permanente, que perdura pelo tempo ou se a falta é Transiente.

Essas classes podem ser combinadas entre si, mas algumas combinações não podem ocorrer por justamente serem antagônicas, como, por exemplo, uma falta maliciosa não pode ser combinada como uma falta não deliberada, pois para causar

dano a um sistema esta foi planejada anteriormente. O Quadro 6-4 define todas as classes e subitens.

Quadro 6-4. Classes da origem de falhas (AVIZIENIS ET AL., 2004)

Classe	SubItem
Fase de Criação	Fase de Desenvolvimento
	Fase Operacional
Fronteiras do Sistema	Falta Interna
	Falta Externa
Causa Fenomenológica	Falta Natural
	Falta por Ato Humano
Dimensão	Falta por Hardware
	Falta por Software
Objetivo	Falta Maliciosa
	Falta Não Maliciosa
Intenção	Falta Deliberada
	Falta Não-Deliberada
Capacidade	Falta Acidental
	Falta por Incompetência
Persistência	Falta Permanente
	Falta Transiente

Domínio de Falhas

De acordo com Avizienis et al. (2004) o domínio de falhas pode ser caracterizado através de dois pontos distintos. O primeiro é com relação a falhas de conteúdo, ou seja, o conteúdo da informação que foi entregue ao serviço via interface desvia do conteúdo necessário pela função do sistema.

O segundo é uma falha de tempo, ou seja, o tempo que a informação precisa para ser entregue ao serviço via interface desvia do tempo necessário pela função do sistema, sendo antes ou depois do momento necessário.

Esses dois pontos podem ser combinados, gerando assim uma falta por tempo e conteúdo. Essa combinação pode gerar efeitos como o silêncio da funcionalidade, ou seja, o usuário não consegue obter mais nenhum processamento do sistema ou pode trazer informações erradas para o usuário. A Figura 6-4 mostra a estrutura do domínio de falhas

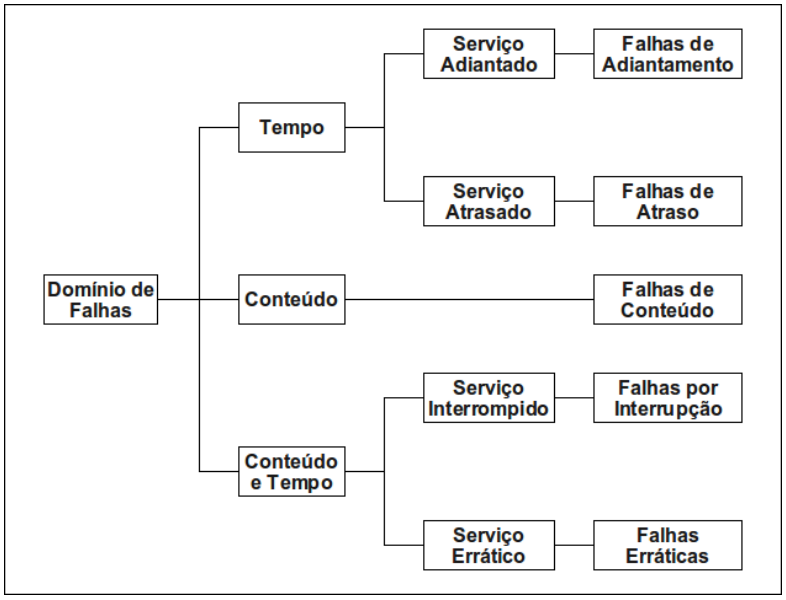

Figura 6-4. Domínio de falhas, adaptado de (AVIZIENIS ET AL., 2004).

Detecção e Manipulação de Erros e Faltas

Avizienis et al. (2004) classifica que um erro é detectado se a sua presença é indicada por mensagens de erro ou sinais de erro no sistema. Erros que estão presentes, mas não foram identificados são chamados de erros latentes. Normalmente ao gerar um erro este quebra uma política de segurança.

Nist (1995) define que o conceito de política de segurança é a de um conjunto de motivações e restrições para segurança, que devem ser respeitados. Dessa forma um software deve ser construído respeitando as políticas de segurança para obter melhor confiabilidade na proteção das funções evitando assim erros e falhas.

Ao criar uma política de segurança um aspecto importante a ser planejado está no processo de detecção e correção de vulnerabilidades (faltas, erros e falhas) nos softwares. Uma seleção de perguntas para a elaboração da política foi construída por APPSIC (2006), dividindo em dois grupos, a identificação e correção de vulnerabilidades e a comunicação e instalação de vulnerabilidades:

- Identificação e correção de vulnerabilidades:
    - Existe uma equipe dedicada para avaliação e suporte sobre as vulnerabilidades de segurança reportadas em seus produtos?
    - Qual é o processo para resolução das vulnerabilidades?
    - Existe uma classificação da gravidade das vulnerabilidades? Como ela é determinada?

    - Existe um acompanhamento das últimas tendências de ataque a software? Como essas afetam os produtos?
    - Existe a publicação de correções para todas as versões suportadas de um produto ao mesmo tempo?
    - Quais são os termos e prazos do contrato de suporte de segurança dos produtos?
    - Quais são as melhorias no processo realizadas devido aos resultados das vulnerabilidades relatadas nos produtos?
- Comunicação e instalação de correções de vulnerabilidades:
    - Existe um canal de comunicação com os clientes para duvidas sobre vulnerabilidades e como estas afetam o software adquirido?
    - Quais são os métodos utilizados para informar os clientes sobre as vulnerabilidades?
    - São fornecidas orientações técnicas sobre vulnerabilidades? Estas podem ser exploradas por terceiros? Como atenuá-las?
    - Qual é a estratégia de lançamento da correção do produto?
    - São fornecidas orientações para implantação / manutenção segura? Há ferramentas de integração para a implantação da correção?
    - Todas as vulnerabilidades que afetam seu software são divulgadas?

Quando ocorre um erro existem diversas técnicas para monitorar ou tratar. O primeiro tipo é a detecção de erros latentes ou faltas dormentes, podendo ser concorrente, ou seja, durante a execução do software, ou preemptiva, durante a suspensão da execução do software.

Outra técnica é a recuperação, ou seja, a transformação do estado de um sistema que contém um ou mais erros e possíveis faltas em um estado sem erros detectados e sem faltas que possam ser ativadas, voltando o sistema a um estado amigável para a utilização do usuário. Dentro dessa técnica existem duas classes de tratamento a de erros e a de faltas. O primeiro consiste na eliminação de erros presentes dentro do estado de falhas do sistema. O segundo consiste na prevenção de faltas para que estas não fiquem ativas novamente.

Avizienis et al. (2004) cita que dentro da classe de tratamento de erros existem três técnicas principais que são utilizadas. O *Rollback* consiste no retorno do sistema a um estado salvo anteriormente, chamado de *checkpoint*, em que o erro ainda não estava ocorrendo. O *Rollforward* consiste na criação de um novo estado

para o sistema sem erros. A Compensation consiste em quando o estado errôneo contem redundância suficiente para mascarar o erro, não mostrando o mesmo para o usuário. A técnica de *Rollback* e *Rollforward* não são mutuamente exclusivas, pois pode haver uma redundância utilizando a técnica de *Rollback*, voltando a um ponto onde não havia erros, mas caso persista, utiliza-se a técnica de *Rollforward* levando o sistema a um estado novo, sem erros.

Avizienis et al. (2004) cita que dentro da classe de tratamento de faltas existem quatro técnicas principais. A primeira é a de Diagnostico que consiste em localizar e identificar os itens que estão causando as faltas. A segunda é o Isolamento onde existe um processo de exclusão lógica ou física dos componentes com faltas para evitar sua participação no sistema, fazendo com que a falta permaneça dormente. A terceira é a Reconfiguração que é a troca de componentes defeituosos por componentes sem faltas ativas. A quarta é a Reinicialização que modifica, atualiza e registra uma nova configuração sobre as tabelas e registros de atualizações.

Chowdhury e Zulkernine (2011) descrevem uma relação de vulnerabilidades em componentes, analisando o seu código. Para o trabalho eles utilizaram métricas de software, principalmente complexidade, acoplamento e coesão, pois estas podem ser avaliadas em qualquer fase do ciclo de vida. Foram mapeados os arquivos do navegador Mozilla Firefox marcando quais possuem correção de vulnerabilidades e foram aplicadas as métricas com a ajuda do software WEKA (2011) e quatro cálculos estatísticos do software: a Arvore de Decisão C4.5 (Decision Tree), Florestas Aleatórias (Random Forests), Regressão Logística (Logistic Regression) e o Naïve-Bayes. Outros trabalhos também focaram os aspectos relacionados com as vulnerabilidades de componentes, como os de Neuhaus et al. (2007), Shin e Williams (2008) e (2008a).

Modelagem de Ameaças

A modelagem de ameaças é um processo formal para identificar, documentar e mitigar ameaças de segurança a um sistema de software. Além disso, ajuda times de desenvolvedores a entender a forma de trabalho de ameaças, observando o sistema a partir dos olhos de um potencial adversário (OLADIMEJI ET AL., 2006).

Dessa forma é necessário investigar todos os pontos de um software para modelar as ameaças que o afetam, sendo muito mais prático e eficaz a utilização de

ferramentas para a modelagem de ameaças. O Microsoft STRIDE/DREAD *model* (2012) define uma série de ameaças ajudando a calcular os riscos existentes dentro do software. Essa modelagem possui seis categorias gerais:

- **Spoffing**: Tentativa de obter acesso a um sistema usando um usuário falso ou uma credencial falsa ou roubada. Após o ingresso do usuário falso no sistema, este pode evoluir o nível de seus ataques.
- **Tampering**: Manipulação e modificação de dados de forma não autorizada.
- **Repudiation**: Característica de o usuário negar a realização de uma ação ou transação. O usuário pode não ser legítimo e para obter uma prova contrária é necessário um bom trabalho de auditoria.
- **Information Disclosure**: Divulgação de informações privadas sem autorização do proprietário.
- **Denial of Service**: Processo de deixar um software indisponível. Este ataque pode ser realizado enviando muitas requisições para uma entrada do software, consumindo todos os recursos causando uma parada total.
- **Elevation of Privileges**: Processo de assumir a identidade de um usuário com mais privilégios, a partir de um usuário com poucos privilégios.

Além dessas categorias gerais de ameaças foi criada uma metodologia para calculo de riscos, DREAD (2012), baseado em cinco itens:

- **Damage potencial**: Esse item caracteriza o quão alto é o dano ao sistema se a vulnerabilidade for explorada.
- **Reproducibility**: Esse item caracteriza o quão fácil é a reprodução do ataque.
- **Exploitability**: Esse item caracteriza o quão fácil é a execução do ataque.
- **Affected users**: Esse item caracteriza uma estimativa de usuários afetados pela vulnerabilidade.
- **Discoverability**: Esse item caracteriza o quão fácil é a descoberta da vulnerabilidade.

Outros exemplos de modelagem de ameaças podem ser encontrados nas ferramentas *Trike* (2012), *Common Vulnerability Scoring System*, CVSS, (2012) e o *Operationally Critical Threat, Asset, and Vulnerability Evaluation*, OCTAVE, (2012).

APÊNDICE E – Tópicos em Manipulação de Dados de Entrada

Estratégia de Validação de Dados

De acordo com OWASP (2012) existem quatro formas para criar uma estratégia de validação de dados: A validação inexistente, as listas brancas, as listas negras e a limpeza de dados.

A estratégia de validação inexistente é totalmente insegura e fortemente desencorajada, pois deixa a aplicação vulnerável.

A estratégia de listas brancas consiste na ideia de criar um conjunto de restrições para valores bons conhecidos. A principal característica é rejeitar todos os dados que não estejam de acordo com esse conjunto de restrições. Alguns exemplos de restrições podem ser:

- Todas as variáveis devem ser fortemente tipadas
- O tamanho dos campos devem ser minimizados e checados
- Deve ser checada a faixa de valores se o dado for numérico
- A sintaxe ou gramática deve ser verificada antes do primeiro uso ou inspeção
- Se for esperado um código postal, valide um código postal (tipo, tamanho e sintaxe)

A estratégia de listas negras é mais fraca do que as listas brancas. Essencialmente se não é esperado ver determinados conjuntos de caracteres em uma entrada, então os rejeite. Mas é uma estratégia perigosa porque o conjunto de possíveis dados ruins é potencialmente infinito. Adotar esta estratégia significa que a manutenção desse conjunto deverá ser realizada eternamente, já que novos dados maliciosos podem surgir todos os dias, tendo dessa forma uma proteção incompleta.

A estratégia de limpeza de dados consiste em modificar a entrada do usuário em um formato aceito. Essa modificação pode ser realizada utilizando listas brancas ou listas negras. Para a limpeza com listas brancas, a estratégia deve ser remover ou substituir caracteres que não estejam no conjunto de restrições. Por exemplo, se for necessário um número de telefone, remova todos os caracteres não numéricos.

Para a limpeza utilizando listas negras, a estratégia deve ser remover ou substituir caracteres rejeitados, em um esforço de transformar o dado de entrada em algo seguro. Essa estratégia sofre dos mesmos problemas de manutenção que a própria estratégia de listas negras.

Estratégias de Defesa

É necessário criar mecanismos para facilitar a validação dos dados de entrada isso é chamado estratégia de defesa. Howard e LeBlanc (2002) exemplificam essa estratégia com dois itens principais. A definição de um limite de confiança ao aplicativo e a criação de pontos de estrangulamento para as entradas.

O primeiro item define que todos os softwares possuem uma camada em que se acredita que os dados estão bem formados e seguros, por validações anteriores. Assim não existe razão para que esses dados sejam novamente verificados. Esse é o chamado principio de defesa em profundidade, pois se caso uma camada for comprometida, existem outras que garantem a segurança.

O segundo item consiste em criar pontos de entrada, chamados de pontos de estrangulamento, para controlar a entrada de dados dentro do limite de confiança do sistema. Não existe uma restrição para a quantidade de pontos, contanto que todos validem os dados de entrada, evitando problemas. Dessa forma é perceptível que o limite de confiança e os pontos de estrangulamento estão intimamente ligados.

Internacionalização de Dados

Um equivoco que ocorre com frequência e serve como porta de ataque está nas questões de internacionalização, pois a maioria dos arquitetos planeja o software sem a validação para conjuntos de caracteres Unicode. O termo dado a esse internacionalização do idioma é I18N que significa em inglês a palavra *internacionalization*, ou seja, o caractere I seguindo de 18 caracteres e finalizando com o caractere N (HOWARD; LEBLANC, 2002).

Os mesmos autores definem duas regras de ouro para a segurança em questões de internacionalização. “A primeira é a utilização do Unicode para a construção do aplicativo e a segunda é a não conversão entre Unicode e outros conjuntos de caracteres/páginas de código.” O autor explica que existem três representações binárias de codificação Unicode: UTF-8, UTF-16 e UTF-32. A principal forma suportada pelos sistemas operacionais atualmente é o UTF-16, mas várias páginas da web realizam o suporte ao UTF-8. Mesmo com essas representações de conjuntos de caracteres, existem múltiplas representações binárias para uma única string, dificultando operações como indexação e validação, aumentando a complexidade e comprometendo a segurança.

Expressões Regulares

Para uma validação de dados simples, podem ser utilizados códigos baseados em comparações de strings simples. Entretanto para dados complexos é necessário utilizar de construções de nível mais alto, como expressões regulares, que são textos especiais para descrever um determinado padrão (OWASP, 2012).

De acordo com Howard e LeBlanc (2002) expressões regulares servem a dois propósitos principais. Encontrar dados em uma determinada String. E realizar a validação de dados. Para clarear é necessário pensar que a string validada contém um nome de arquivo, o objetivo não é encontrar o nome do arquivo, mas determinar que o nome de arquivo seja válido.

Uma importante forma utilizar expressões regulares é a validação de dados internacionais. Normalmente estas lidavam apenas com caracteres de oito bits, ou seja, valores pertencentes a um código latino com representações da tabela básica de ASCII. É necessário melhorar as técnicas de validação entre expressões e o Unicode, pois há poucos mecanismos que suportam corretamente o Unicode.

Outro ponto relacionado a expressões regulares e idiomas representados pelo Unicode é que os idiomas são entidades que evoluem ao longo do tempo. Um caractere que hoje é inválido em um idioma pode se tornar válido no dia seguinte.

Formatos e Vulnerabilidades

Existem alguns padrões para estabelecer formatos dos mais variados tipos de dados. Para data e hora pode ser consultado o padrão ISO 8601 (ISO/IEC, 2004). Para a parte de dados primitivos e itens da linguagem como inteiros, strings, números com ponto flutuante a melhor forma de obter informações sobre o formato é pesquisar na documentação da linguagem de programação. Ainda podem ser utilizados os padrões de formato de outros itens relacionados à segurança como a parte de biometria de impressões digitais definida na ISO 19794 (ISO/IEC, 2011a).

Algumas das vulnerabilidades conhecidas sobre os dados de entrada são o *SQL Injection* e o *Cross-site Scripting*. Ambas são entradas mal intencionadas, onde o primeiro é uma entrada sobre um banco de dados, ao modificar comandos de linguagem SQL (*Structured Query Language*) para inserir, modificar, apagar ou obter informações. O segundo age sobre uma requisição HTTP redirecionando o usuário para outro site que o atacante deseja.

Para facilitar o trabalho de validação dos dados de entrada, existem aplicações para esse fim. Alguns exemplos de aplicações são o *Commons Validator* (2012) e o *OWASP Stinger Project* (2012).

O *Commons Validator* é um componente de validação de código aberto que teve origem no framework *Apache Struts*. O alvo de validação são os *JavaBeans*. As regras são chamadas de *validator actions* e são métodos estáticos de retorno booleano dentro dos objetos java. O componente possibilita aos desenvolvedores configurar métodos alvo e regras dentro dos arquivos de validação.

O *OWASP Stinger Project* é um validador de requisições HTTP para ser usado em um ambiente Java Enterprise. Esse suporta regras sobre expressões regulares e para a falta ou excesso de itens da requisição HTTP. Desenvolvedores podem especificar as regras utilizando o Security Validation Description Language.

Para finalizar segue a lista de benefícios da validação de dados de entrada:

- Pode prevenir ataques que estejam dentro do conteúdo da mensagem
- Listas brancas podem afastar novos tipos de ataque desconhecidos que estão fora do formato padrão dos validadores
- O mecanismo habilita uma contabilidade dos ataques de violação de regras, sendo que esta informação pode ser utilizada para reconstruir brechas de segurança.
- O validador cria um gerenciável mecanismo central

A mesma regra de validação pode ser aplicada em diferentes alvos, realizando um reúso da lógica de validação.

APÊNDICE F – Pesquisa Acadêmica

Pesquisa Acadêmica

Esta pesquisa tem como objetivo realizar a avaliação de um Modelo de Qualidade para Características de Segurança de um Componente de Software. Esta é uma das etapas da dissertação de mestrado de Everson Carlos Mauda, que está sendo desenvolvida sob orientação da Dra. Sheila Reinehr, no Programa de Pós-Graduação em Informática (PPGIa) da Escola Politécnica da Pontifícia Universidade Católica do Paraná (PUCPR). O título deste trabalho é: “Modelo de Qualidade de Características Internas de Segurança para Componentes de Software”. Desde já agradeço pela sua atenção e colaboração.

Everson Carlos Mauda (maudaeverson@gmail.com)
Curitiba 2 de Julho de 2012

Esclarecimentos sobre a pesquisa:

O tempo total previsto para responder esta pesquisa é de até 20 minutos. A pesquisa está organizada da seguinte forma:

- **Parte 1** – Caracterização do Profissional
- **Parte 2** – Características das áreas de segurança para produto de software
- **Parte 3** – Modelo de qualidade de características internas de segurança de um componente de software
- **Apêndice A** – Acordo de Confidencialidade
- **Apêndice B** – Modelo de Qualidade --> Por restrições do google docs o modelo se encontra em meu site pessoal em separado: http://www.mauda.com.br/paginas/modelo.jsf#modelo

Parte 1 - Caracterização do Profissional

Esta seção visa obter informações de modo a caracterizar o respondente.

1.1 – Nome:

Digite seu nome completo

_____

1.2 – Email:

Digite seu email

_____

1.3 – Nome da Instituição/Empresa

Digite o nome da Instituição/Empresa que você atua

_____

1.4 – Áreas de Atuação

Defina o seu nível de experiência nas seguintes áreas

Área	Nunca Atuei	Até 2 anos	Até 5 anos	Até 8 anos	Mais de 8 anos
Engenharia de requisitos					
Análise e projeto de sistemas					
Arquitetura de sistemas					
Arquitetura de segurança					
Implementação					
Testes					
Implantação					
Treinamento corporativo					
Ensino e pesquisa acadêmica					

1.5 – Experiência em componentes de software:

Defina o seu nível de experiência nas áreas relacionadas a componentes de software

Área	Nunca Atuei	Até 2 anos	Até 5 anos	Até 8 anos	Mais de 8 anos
Desenvolvimento de componentes de software					
Ensino e pesquisa acadêmica em componentes de software					

1.6 - Você já desenvolveu algum trabalho acadêmico relacionado com a área de Segurança de Software:

Esta área refere-se a itens de segurança relacionados com o produto de software:

[ ] Nunca desenvolvi nenhum trabalho acadêmico nesta área
[ ] Trabalho de conclusão de curso de graduação
[ ] Monografia de especialização
[ ] Dissertação de mestrado
[ ] Tese de doutorado
[ ] Publicação de artigos
[ ] Pesquisa própria
[ ] Outro: _____

1.7 - Você já orientou algum trabalho acadêmico relacionado com a área de Segurança de Software:

Esta área refere-se a itens de segurança relacionados com o produto de software

[ ] Nunca orientei nenhum trabalho acadêmico nesta área
[ ] Trabalho de conclusão de curso de graduação
[ ] Monografia de especialização
[ ] Dissertação de mestrado
[ ] Tese de doutorado
[ ] Outro: _____

1.8 - Você já desenvolveu algum trabalho acadêmico relacionado com a área de Segurança da Informação:

Esta área refere-se a itens de segurança relacionados com o hardware e servidores de aplicação

[ ] Nunca desenvolvi nenhum trabalho acadêmico nesta área
[ ] Trabalho de conclusão de curso de graduação
[ ] Monografia de especialização
[ ] Dissertação de mestrado
[ ] Tese de doutorado
[ ] Publicação de artigos
[ ] Pesquisa própria
[ ] Outro: _____

1.9 - Você já orientou algum trabalho acadêmico relacionado com a área de Segurança da Informação:

Esta área refere-se a itens de segurança relacionados com o hardware e servidores de aplicação

[ ] Nunca orientei nenhum trabalho acadêmico nesta área
[ ] Trabalho de conclusão de curso de graduação
[ ] Monografia de especialização
[ ] Dissertação de mestrado
[ ] Tese de doutorado
[ ] Outro: _____

1.10 - Certificações na Área de Segurança:

Marque as certificações que você possui, caso não o tenha marque o item "não possuo certificação na área de segurança"

[ ] Certified Ethical Hacker (CEH)
[ ] Certified Expert Penetration Tester (CEPT)
[ ] Certified Information Security Manager (CISM)
[ ] Certified Information Systems Auditor (CISA)
[ ] Certified Information Systems Security Professional (CISSP)
[ ] Certified Penetration Tester (CPT)
[ ] Certified Secure Software Lifecycle Professional (CSSLP)
[ ] Certified Wireless Security Professional (CWSP)
[ ] CheckPoint Certified Security Administrator (CCSA)
[ ] CheckPoint Certified Security Expert (CCSE)
[ ] CyberSecurity Forensic Analyst (CSFA)
[ ] EC-Council/Certified Hacking Forensic Investigator (CHFI)
[ ] GIAC Certified Intrusion Analyst (GCIA)
[ ] GIAC Certified Windows Security Administrator (GCWN)
[ ] GIAC Secure Software Programmer - .NET (GSSP-NET)
[ ] GIAC Secure Software Programmer - Java (GSSP-JAVA)
[ ] Não possuo certificação na Área de Segurança
[ ] Outro: _____

Parte 2 - Características das áreas de segurança para produto de software

Esta seção aborda aspectos relacionados às áreas de segurança e suas características principais.

2.1 - Em sua opinião, qual é a importância das seguintes áreas de segurança:

Área	Não é importante	Baixa	Intermediária	Alta	Imprescindível
Auditabilidade					
Autenticação					
Criptografia de dados					
Controle de vulnerabilidades					
Manipulação de dados					

2.2 - Em sua opinião, existe alguma área de segurança que não foi mencionada? Como você classifica a importância desta área?

_____

_____

2.3 - Em sua opinião, qual é a importância das seguintes características referentes à área de Auditabilidade:

Característica	Não é importante	Baixa	Intermediária	Alta	Imprescindível
Classe de eventos (i. e. ERROR, WARNING)					
Padrão de formato do arquivo de log					
Armazenamento do arquivo de log					
Integridade de arquivo de log					
Arquivamento de arquivo de log					
Operação de rotação de arquivo de log					
Compressão de arquivo de log					
Visualização de arquivo de log					
Filtragem de eventos de arquivo de log					
Agregação de eventos de arquivo de log					
Correlação de eventos de arquivo de log					
Conversão de arquivo de log					
Normalização de arquivo de log					
Operação de relatório de arquivo de log					

2.4 - Em sua opinião, existe alguma característica da área de Auditabilidade que não foi mencionada acima? Como você classifica a importância desta característica?

_____

_____

2.5 - Em sua opinião, qual é a importância das seguintes características referentes à área de Autenticação:

Característica	Não é importante	Baixa	Intermediária	Alta	Imprescindível
Algoritmos Hash					
Forma de Armazenamento					
Algo que você conhece (Senhas)					
Algo de Posse (Tokens)					
Algo Inerente (Digitais)					
Chaves Criptográficas Públicas					

2.6 - Em sua opinião, existe alguma característica da área de Autenticação que não foi mencionada acima? Como você classifica a importância desta característica?

_____

_____

2.7 - Em sua opinião, qual é a importância das seguintes características referentes à área de Criptografia de Dados

Característica	Não é importante	Baixa	Intermediária	Alta	Imprescindível
Algoritmos simétricos					
Algoritmos assimétricos					
Tamanho da chave criptográfica					
Vida útil da chave criptográfica					
Custo computacional para produzir a chave criptográfica					
Custo computacional para o uso da chave criptográfica					
Armazenamento da chave criptográfica					
Verificação de integridade da chave criptográfica					

2.8 - Em sua opinião, existe alguma característica da área de Criptografia de Dados que não foi mencionada acima? Como você classifica a importância desta característica?

_____

_____

2.9 - Em sua opinião, qual é a importância das seguintes características referentes à área de Controle de Vulnerabilidades

Característica	Não é importante	Baixa	Intermediária	Alta	Imprescindível
Tipo de Falhas					
Origem de Falhas					
Detecção de Falhas					
Manipulação de Erros					
Processo de Modelagem de Ameaças					

2.10 - Em sua opinião, existe alguma característica da área de Controle de Vulnerabilidades que não foi mencionada acima? Como você classifica a importância desta característica?

_____

_____

2.11 - Em sua opinião, qual é a importância das seguintes características referentes à área de Manipulação de Dados

Característica	Não é importante	Baixa	Intermediária	Alta	Imprescindível
Estratégia de Defesa					
Estratégia de Validação					
Algoritmo Hash					
Utilização de Expressões Regulares					
Validação de Dados Internacionais (Unicode)					
Formato dos Dados					

2.12 - Em sua opinião, existe alguma característica da área de Manipulação de Dados que não foi mencionada acima? Como você classifica a importância desta característica?

_____

_____

2.13 - Em sua opinião, qual é o nível da segurança oferecido por cada um dos seguintes itens de segurança:

Característica	Nenhum	Baixo	Intermediário	Alto	Muito Alto
Assinaturas e forma de escrita					
Cartões com Chip + Senha					
Cartões RFID					
Certificados Digitais					
Identificação por Desafio-Resposta					
Impressões Digitais					
Iris					
Passphrases					
Senhas					
Senhas Únicas					

2.14 - Em sua opinião, existe algum outro item de segurança que não foi mencionado acima e que você considera importante? Qual?

_____

_____

2.15 - Caso você tenha inserido uma área de segurança que não foi citada na questão 2.2 ("Em sua opinião, existe alguma área de segurança que não foi mencionada acima? Como você classifica a importância desta área?"), por favor, insira quais são suas características importantes:

_____

_____

_____

_____

Parte 3 - Modelo de qualidade de características internas de segurança de um componente de software

Esta seção refere-se ao Modelo de Qualidade proposto. Por restrições do Google Docs (usado para viabilizar este questionário), não é possível a inserção de figuras. Para ver o modelo completo, por favor, clique em http://www.mauda.com.br/paginas/modelo.jsf#modelo

3.1 Como você avalia a importância do modelo de qualidade que está sendo proposto?

Tenha em mente que o modelo proposto visa auxiliar arquitetos de software e desenvolvedores, que nem sempre possuem grande vivência na área de segurança, a desenvolver componentes mais seguros

_____

_____

3.2 Em sua opinião, as características levantadas no modelo são importantes para a criação de componentes de software com uma maior segurança interna? Por quê? Caso mais alguma característica seja importante, qual seria essa e o por que dessa importância?

Lembrando que as características levantadas no modelo são Criptografia de Dados, Autenticação, Auditabilidade, Controle de Vulnerabilidades, Validação de Dados

_____

_____

3.3 Em sua opinião, as subcaracterísticas do modelo estão com uma boa aglutinação? Existem outras subcaracterísticas que poderiam estar no modelo? Existe alguma subcaracterística que poderia se tornar uma característica?

Lembrando que as subcaracterísticas encontram-se na segunda coluna do modelo

_____

_____

3.4 Em sua opinião, os atributos do modelo estão com uma boa aglutinação? Existe algum atributo que não se encontra no modelo? Existe algum atributo que pode ser excluído do modelo? Qual é a utilidade desses atributos no dia a dia de um projeto de software?

Lembrando que os atributos encontram-se na terceira coluna do modelo

_____

_____

Apêndice A – Termo de Confidencialidade

Este Termo de Confidencialidade visa estabelecer um acordo entre os pesquisadores Everson Carlos Mauda e Sheila Reinehr, doravante denominados Pesquisadores e o profissional de informática doravante denominado Participante, a respeito da confidencialidade das informações coletadas durante o processo de pesquisa da dissertação de mestrado do primeiro, sob orientação do segundo, intitulado: "MODELO DE QUALIDADE PARA CARACTERÍSTICAS INTERNAS DE SEGURANÇA DE COMPONENTES DE SOFTWARE".
Por meio deste Termo de Confidencialidade, os Pesquisadores se comprometem a:

- Portar-se com discrição em todos os momentos da pesquisa acadêmica, não comentando ou divulgando qualquer tipo de informação que tenha sido repassada de forma oral ou escrita;
- Não divulgar o nome do Participante, em qualquer meio, a menos que expressamente autorizado por este;
- Não divulgar, em qualquer meio, os dados e informações individualizadas coletados durante o processo de pesquisa com o Participante;
- Divulgar, em formato de dissertação, artigos e apresentações, apenas os dados agregados, dos quais não se possa retirar ou inferir a identificação do Participante;

As assinaturas abaixo expressam a concordância quanto ao cumprimento deste Termo de Confidencialidade, por prazo indeterminado.

_____ _____

Everson Carlos Mauda *Sheila Reinehr*

Curitiba, 9 de julho de 2012.

Apêndice B – Modelo de Qualidade

Por restrições do google docs por não poder inserir imagens dentro de um questionário, o modelo se encontra em meu site pessoal em separado: http://www.mauda.com.br/paginas/modelo.jsf#modelo