LUIS GONZAGA DE PAULO

UM MODELO COMPLEMENTAR PARA APRIMORAR A SEGURANÇA DA INFORMAÇÃO NO SDLC PARA DISPOSITIVOS MÓVEIS: SDD - *SECURITY DRIVEN DEVELOPMENT*

Curitiba PR

Agosto de 2015

LUIS GONZAGA DE PAULO

UM MODELO COMPLEMENTAR PARA APRIMORAR A SEGURANÇA DA INFORMAÇÃO NO SDLC PARA DISPOSITIVOS MÓVEIS: SDD - *SECURITY DRIVEN DEVELOPMENT*

Dissertação submetida ao Programa de Pós-Graduação em Computação Aplicada da Universidade Tecnológica Federal do Paraná como requisito parcial para a obtenção do título de Mestre em Computação Aplicada.

Área de concentração: *Engenharia de Sistemas Computacionais*

Orientador: Luiz Nacamura Júnior

Co-orientadora: Maria Cláudia F. P. Emer

Curitiba PR

Agosto de 2015

Dados Internacionais de Catalogação na Publicação

P331m 2015

Paulo, Luis Gonzaga de
Um modelo complementar para aprimorar a segurança da informação no SDLC para dispositivos móveis : SDD : security driven development / Luis Gonzaga de Paulo.-- 2015.
210 p. : il.; 30 cm

Texto em português, com resumo em inglês
Dissertação (Mestrado) - Universidade Tecnológica Federal do Paraná. Programa de Pós-graduação em Computação Aplicada, Curitiba, 2015
Bibliografia: p. 114-118

1. Dispositivos móveis. 2. Software - Desenvolvimento. 3. Software - Proteção. 4. Proteção de dados. 5. Hackers. 6. Computação - Dissertações. I. Nacamura Júnior, Luiz, orient. II. Emer, Maria Cláudia Figueiredo Pereira, coorient. III. Universidade Tecnológica Federal do Paraná. Programa de Pós-gradu-ação em Computação Aplicada. IV. Título.

CDD: Ed. 22 -- 621.39

Biblioteca Central da UTFPR, Câmpus Curitiba

ATA DE DEFESA DE DISSERTAÇÃO DE MESTRADO Nº 31

Aos 20 dias do mês de agosto de 2015 realizou-se na sala B-205 a sessão pública de Defesa da Dissertação de Mestrado intitulada "Um Modelo Complementar para aprimorar a Segurança da Informação no SDLC para Dispositivos Móveis: SDD – Security Driven Development", apresentada pelo aluno **Luis Gonzaga de Paulo** como requisito parcial para a obtenção do título de Mestre em Computação Aplicada, na área de concentração "Engenharia de Sistemas Computacionais", linha de pesquisa "Sistemas de Informação".

Constituição da Banca Examinadora:

Prof. Dr. Luiz Nacamura Júnior, UTFPR - CT (Presidente) _____

Prof. Dr. Carlos Alberto Maziero, UFPR _____

Prof. Dr. Altair Olivo Santin, PUC - PR _____

Em conformidade com os regulamentos do Programa de Pós-Graduação em Computação aplicada e da Universidade Tecnológica Federal do Paraná, o trabalho apresentado foi considerado _____ (aprovado/reprovado) pela banca examinadora. No caso de aprovação, a mesma está condicionada ao cumprimento integral das exigências da banca examinadora, registradas no verso desta ata, da entrega da versão final da dissertação em conformidade com as normas da UTFPR e da entrega da documentação necessária à elaboração do diploma, em até _____ dias desta data.

Ciente (assinatura do aluno): _____

(para uso da coordenação)

A Coordenação do PPGCA/UTFPR declara que foram cumpridos todos os requisitos exigidos pelo programa para a obtenção do título de Mestre.

Curitiba PR, _____/_____/_____ _____

"A Ata de Defesa original está arquivada na Secretaria do PPGCA".

À minha amada Rogéria Reguim de Paulo, à minha filha Amanda Cristina e aos meus filhos André Luís, Luís Otávio e João Pedro, que são meu maior tesouro aqui na terra.

A meu pai, Antônio Francisco de Paulo, à saudosa memória de minha mãe, Maria José Vilela de Paulo, e de meus avós José "Menino" e Maria Vitória.

Foram, são e sempre serão, todos, incansáveis incentivadores, e os principais responsáveis por todos os meus triunfos.

Agradecimentos

Primeiramente e principalmente a Deus, Autor e Senhor de minha vida.

A meus pais, Antônio Francisco de Paulo e Maria José Vilela de Paulo (*in memoriam*), meus primeiros e eternos mestres.

Ao meu orientador, Profº. Drº. Nacamura, e a minha co-orientadora, Profª. Maria Cláudia, pela firmeza e determinação, pelo zelo para com a qualidade do trabalho, pela compreensão e principalmente pela confiança em mim depositada, e a todos os professores do PPGCA/DAINF, em especial aos professores Adolfo e Maziero, pela indispensável ajuda e apoio nos momentos mais críticos.

Aos meus inúmeros e valorosos mestres – especialmente à Zaíra Tribst, Renato, Armindo Paione, Otaviano Pereira, do "Industrial". À Targino Nogueira, Wanderlei Bueno e Carson, da ETEV, e aos sábios Aida Pita Pinto, Júlio Albertini, Kristhian Gregersen, Fábio Feijó, Atanase Nicolas Gatos, entre tantos outros com quem tive o prazer e a honra de estudar, trabalhar e conviver, e que me contaminaram com o gosto pela ciência, pela cultura, pelas artes, pela filosofia e pela pesquisa.

Aos amigos Profºs Hugo Neto, Gilson Sato, Ubiradir, Kleber, Foronda, Paulo Absy, Edson Ferlin, Cláudio Oliveira, e aos colegas, professores e alunos da UTFPR e da UNINTER, pelo apoio no início da jornada, pelo incentivo, pela constante ajuda e pelos muitos momentos de descontração e alegria.

Aos gestores e colegas das empresas BRQ, NTTDATA e WIPRO, pelo incentivo, pela compreensão, tolerância e apoio nas horas mais críticas.

Aos parentes, amigos e colegas de trabalho, dos quais suprimi valiosos momentos de companhia e descontração para dedicar-me ao presente trabalho.

Que Deus os abençoe e os recompense, e que saibam que contam com a minha eterna gratidão.

Pan Metros Aristos - O bom é a medida do melhor (Provérbio Grego).

Omnia Tempus Habent – Para tudo há um tempo (Ec. 3,1).

Resumo

O uso de dispositivos móveis por um número cada vez maior de pessoas, e em um número crescente de atividades que requerem mais segurança da informação, coloca em evidência a necessidade de prover segurança nos softwares desse ambiente. O aspecto de segurança da informação em dispositivos móveis é preocupante. Entretanto os modelos utilizados pela indústria de software – e os encontrados na literatura atual - no desenvolvimento de aplicações móveis com requisitos de segurança da informação de alto nível ainda não respondem às necessidades de mais segurança reclamadas pelos usuários. O presente estudo considera que tais modelos podem ser melhorados com o incremento de métodos e técnicas específicas, algumas já utilizadas com sucesso no desenvolvimento de aplicações desktop ou não voltadas para o ambiente de dispositivos móveis. Este trabalho propõe a inclusão de abordagem de segurança da informação no início do ciclo de vida do desenvolvimento de software, a partir do estudo das ameaças e vulnerabilidades, da aplicação antecipada dos casos de abuso – aqui chamados de casos de uso impróprio, da análise de risco, dos testes de segurança baseados no risco e do uso de máquinas de ataque nos testes de segurança durante o processo de desenvolvimento do software. Para alcançar o objetivo desta pesquisa, os modelos mais conhecidos e utilizados no ciclo de vida do desenvolvimento de software são analisados do ponto de vista da segurança da informação, e uma nova abordagem é proposta por meio do uso de um modelo complementar de desenvolvimento de software voltado para a segurança. Alguns modelos de artefatos são apresentados e um estudo de caso aplicando os conceitos tratados na pesquisa é utilizado com o intuito de avaliar as principais contribuições discutidas no texto, e também alguns dos resultados preliminares obtidos com a realização do trabalho de pesquisa.

Palavras-chave: Segurança da Informação. Engenharia de Software. SDLC. Caso de Abuso. Análise de Risco. Máquina de Ataque. Dispositivos Móveis. Mobile.

Abstract

The increasingly wide and intense use of mobile devices - whose processing and storage capacity grows almost overcoming the desktops - exposes greatly issues relating to information security in this environment. This is a worrying fact. However, the models currently found in the literature and used by software industry in developing mobile applications with the highest information security requirements are not yet answering users' needs for more security, and may be improved adding specific methods or techniques, sometimes already used in desktop - or not mobile ones - applications development. This work proposes to insert information security approach early in the software development life cycle using threats and vulnerabilities study, the early application of abuse case - also called misuse cases, the risk analysis, the risk based security test and the use of attack machines in the development process. To reach the research goal, this work analyzed usual models used on SDLC from the information security point of view, and presents a new approach thru the use of a security driven development complementary model. The work also presents some templates and uses a case study for apply the concepts and evaluate the main contributions discussed in the text, also as the preliminary results obtained on the research.

Keywords: *Software Engineering. Information Security. Software Development Lifecycle. Abuse Case. Misuse Case. Attack Machine. Mobile. Risk Analysis.*

Lista de Figuras

Lista de Tabelas

Lista de Quadros

Lista de Siglas e Abreviações

3G	Telefonia Celular de Terceira Geração.
4G	Telefonia Celular de Quarta Geração.
5G	Telefonia Celular de Quinta Geração.
AIDD	*Attack Identification Description and Defense* ou Descrição da Identificação do Ataque e Defesa.
BDD	*Behavior Driven Development* ou desenvolvimento guiado por comportamento.
BYOD	*Bring Your Own Device* ou traga seu próprio dispositivo.
CBSE	*Component Based Software Engineering* ou Engenharia de Software Baseada em Componentes.
CERT_BR	Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil.
COTS	*Commercial Off-the-Shelf* ou Componentes de Prateleira.
CRM	*Customer Relationship Management* ou Gerenciamento do Relacionamento com o Cliente.
EDGE	*Enhanced Date Rates for GSM Evolution* ou Taxas de Dados Ampliadas para a Evolução do GSM.
EIOD	*Extended Interaction Overview Diagrams* ou Diagramas de Visão Geral de Interação Estendida.
FDD	*Feature Driven Development* ou Desenvolvimento Guiado por Funcionalidades.
FSM	*Finite State Machines* ou Máquinas de Estado Finito.
GRPS	*General Packet Radio Service.*
GSM	*Global System for Mobile Communications* ou Sistema Global para Comunicações Móveis.
HSPA	*High Speed Packet Access* ou Acesso de Alta Velocidade por Pacotes.
IEEE	*Institute of Electrical and Electronics Engineers* ou Instituto de Engenheiros Eletricistas e Eletrônicos.
IOT	*Internet of Things* – Internet das Coisas.
MDA	*Model Driven Architecture* ou arquitetura dirigida por modelo.
MDD	*Model Driven Development* ou desenvolvimento dirigido por modelo.
MDE	*Model Driven Engineering* ou engenharia dirigida por modelo.
MDS	*Model Driven Security* ou segurança dirigida por modelo.
MetaISA	*Meta Instruction Set Architecture.*
MMS	*Multimedia Message System* ou Sistema de Mensagens Multimídia.
MUCSIM	*Misuse Case Simulator* ou simulador de caso de uso impróprio.
NFC	*Near Field Communication.*
OCL	*Object Constraint Language* ou linguagem de objetos de restrições.
OWASP	*Open Web Application Security Project*.
POC	*Proof of Concept* ou Prova de Conceito.
RAD	*Rapid Application Development* ou Desenvolvimento Rápido de Aplicações.
ReTBLDTG	*ReTargetable Binary-Level Dynamic Test Generation*.

ROI	*Return of Investment* ou Retorno do Investimento.
SaaS	*Software as a Service*, ou Software como Serviço.
SCR	*Software Cost Reduction* ou Redução de Custo de Software.
SDD	*Security Driven Development* ou Desenvolvimento Dirigido pela Segurança
SDLC	*Software Development Lifecycle* ou Ciclo de Vida do Desenvolvimento de Software
SEI	*Software Engineering Institute* ou Instituto de Engenharia de Software.
SER	*Security Requirement Engineering* ou Engenharia de Requisitos de Segurança.
SMS	*Short Message Service* ou Serviço de Mensagens Curtas (ou de texto).
SoC	*Separation of Concerns* ou separação entre requisitos de segurança e do negócio.
STRIDE	*Spoofing identity, Tampering with data, Repudiation, Information disclosure, Denial of service, and Elevation of privilegie* ou roubo de identidade, adulteração de dados, repúdio, divulgação de informações, negação de serviço e elevação de privilégio.
TAF	*Test Automation Framework* ou Estrutura de Automação de Testes.
TAF-SFT	*Test Automation Framework – Security Funcional Testing* ou Framework de Automação de Testes – Teste de Segurança Funcional.
TCG	*Trusting Computer Group.*
TCSS	*Trusting Computer Supporting Software* ou Software de Apoio à Computação Confiável.
TDD	*Test Driven Development* ou desenvolvimento guiado por testes.
UIT	*Union Internationale des Télécommunications* ou União Internacional de Telecomunicações.
UMTS	*Universal Mobile Telecommunications System.*
WAP	*Wireless Application Protocol* ou Protocolo para Aplicações sem fio.
Wi-Fi	*Wireless Fidelity.*
WLAN	*Wireless Local Area Network* ou Rede Local sem fio.
XP	*Extreme Programming.*

Sumário

Capítulo 1 - Introdução

A popularização do uso de dispositivos de computação móvel – ou *mobile*, principalmente *smart phones* e *tablets* - e particularmente, no Brasil, o aumento do uso dos serviços de comunicação de dados decorrente da redução de custo – quer seja dos equipamentos quer seja dos serviços, e o aumento da velocidade de acesso disponibilizada pelas prestadoras de serviços de comunicação com as redes 3G e 4G alimentam o mercado, estimulando o crescimento do número de aplicações disponíveis. Entretanto este crescimento não se reflete na conscientização de usuários e desenvolvedores de software quanto aos riscos aos quais a informação está submetida nestes ambientes (Chandramohan; Kuan, 2012).

Os desenvolvedores de software não estão habituados a adotar procedimentos de segurança no desenvolvimento de software, tornando-o vulnerável (Braga et al, 2012), e em um momento no qual tendências como o BYOD – *Bring Your Own Device* – literalmente, traga seu próprio dispositivo, e a IOT – *Internet of Things* – Internet das Coisas - apresentam-se como diferencial competitivo e potencial para viabilizar a comunicação universal e integrada, é necessário reforçar a segurança da informação nos dispositivos móveis e garantir a confiabilidade do software nestes dispositivos, pois tais tendências são irreversíveis (Taurion, 2013).

No Brasil o crescimento reflete-se na ultrapassagem do número de aparelhos ativos sobre a população do país (Ericsson, 2013). Também é expressivo e igualmente notável o crescimento da utilização de dispositivos móveis, especialmente *tablets* e *smart phones*, quer seja para uso pessoal, quer seja para uso profissional, os quais fazem uso da rede privativa de telefonia celular para usufruir dos serviços de telefonia, acesso à internet e troca de mensagens. Neste mesmo viés, constata-se a evolução da capacidade do hardware, das funcionalidades dos sistemas operacionais e dos aplicativos, incluindo-se aí o incremento da preocupação destes com a integridade, disponibilidade e confidencialidade – trinômio fundamental da segurança da informação (Avizienis et al, 2004). Embalada por este crescimento, a oferta de jogos, aplicações de uso pessoal ou social, comunicação, acesso a redes sociais e utilidades em geral vem sistematicamente superando seus recordes.

Infelizmente este cenário de crescimento atrai também mentes maliciosas em busca de lucro fácil ou notoriedade, tornando-se um terreno fértil para a produção e disseminação de *malwares*. Paradoxalmente, a disponibilidade de aplicações de acesso e serviços bancários (o

mobile banking), de ferramentas de comércio eletrônico e de aplicações de uso profissional ou corporativo não expressa o mesmo movimento de crescimento, em parte devido à falta de mão de obra especializada no desenvolvimento, e em parte devido à incerteza quanto aos riscos de segurança neste ambiente (Taurion, 2013).

A preocupação com a segurança da informação é uma questão recorrente na indústria da tecnologia da informação, abrangendo os diversos setores dessa indústria. O crescimento do uso da internet, e mais recentemente a crescente utilização de dispositivos móveis reforçou esta preocupação. Um grande contingente de usuários preocupa-se com a segurança das informações produzidas, armazenadas e tratadas por estes dispositivos de forma mais acentuada que nos ambientes fixos e de maior porte, tais como os desktops e os notebooks. A rigor, a segurança da informação em dispositivos móveis tem ficado circunscrita à identificação das vulnerabilidades nos diferentes subsistemas que compõem estes dispositivos e na criação de mecanismos de proteção ou defesas aos ataques no ambiente operacional, desvinculada do processo de desenvolvimento de software (Chandramohan; Kuan, 2012),

Este trabalho apresenta um modelo complementar ao processo desenvolvimento de software aplicativo – aplicações para uso pessoal e profissional em ambiente de computação móvel. Este modelo, dirigido pela segurança da informação, enfatiza e integra a elaboração de casos de uso impróprio, a análise de riscos, o projeto e a construção de máquinas de ataque e o teste voltado para a segurança da informação aos modelos de desenvolvimento de software utilizados na atualidade. Também propõe formas de identificar e classificar as ameaças à segurança da informação e expandir o conhecimento de usuários, desenvolvedores de software, empresas e instituições acerca dos mecanismos de segurança, bem como das potenciais vulnerabilidades e falhas do sistema operacional dos dispositivos de computação móvel.

O trabalho apresenta um estudo de caso com base no sistema operacional da Google, o Android, o que se deve ao fato de que o seu uso em dispositivos móveis como base para aplicações pessoais e corporativas que requerem segurança – tais como *mobile banking*, comércio eletrônico e comunicação profissional - vem se tornando um imperativo. Ao mesmo tempo, e infelizmente, a facilidade de acesso ao código e o grande número de dispositivos móveis que fazem uso do Android tornam atrativas e rentáveis as atividades criminosas que exploram as vulnerabilidades desse sistema operacional.

1.1 Objetivos

1.1.1 Objetivo geral

Propor um modelo complementar que busca aprimorar a segurança da informação em todo o ciclo de vida dos softwares desenvolvidos para aplicações de dispositivos móveis, empregando a abordagem precoce da segurança, quer seja, incrementar qualquer modelo de desenvolvimento de software para que tenha o foco voltado para a segurança da informação.

1.1.2 Objetivos específicos

O presente trabalho integra objetivos do domínio da segurança da informação aos do domínio da engenharia de software.

Quanto à Segurança da Informação é o propósito deste trabalho:

- Apoiar o estudo, a avaliação e a classificação das vulnerabilidades e ameaças e o impacto destas na segurança da informação das aplicações para os dispositivos móveis;
- Subsidiar a avaliação dos aspectos da segurança da informação durante o processo de desenvolvimento de aplicações para os dispositivos móveis;
- Auxiliar a identificação de boas práticas e a aplicação de recomendações de segurança da informação no ambiente de desenvolvimento;
- Estudar questões relativas ao tratamento da segurança da informação nos softwares desenvolvidos para ambiente de dispositivos móveis;
- Analisar e buscar formas efetivas de promover a integração das equipes de desenvolvimento de software e de segurança da informação durante o processo de desenvolvimento de software;

No que ser refere à engenharia de software, a proposta deste trabalho considera:

- Abordar os conceitos da Engenharia de Software aplicáveis ao SDLC – *Software Development Lifecycle* ou Ciclo de Vida do Desenvolvimento de Software - nos diversos modelos de desenvolvimento praticados pela indústria de software e pelas organizações em geral, e a interdependência entre estes conceitos e a segurança da informação;

- Apresentar um modelo complementar ao processo de desenvolvimento de software, tornando-o dirigido para a segurança da informação, o qual possa integrar-se aos modelos e às práticas vigentes e reforçar os aspectos de segurança em todo o ciclo de vida do software;

1.2 Metodologia

Este trabalho foi desenvolvido com base em pesquisa de artigos, teses, dissertações e monografias nas bases de dados públicas de instituições de ensino como UTFPR, UFPR, PUC-PR, UNICAMP, UNB, instituições de pesquisa e entidades como SBC, ABNT, IEEE, ACM, fabricantes e fornecedores de produtos e serviços para dispositivos móveis, como Google, Ericsson, Apple, Nokia-Siemens, Samsung. Além disso foram realizadas pesquisas em livros impressos e em bibliotecas virtuais, e também em portais, sites e fóruns na Internet.

A pesquisa em questão privilegiou a produção acadêmica do período de 2010 em diante, com exceções para propostas e conceitos fundamentais nas áreas de redes e comunicação, engenharia de software e segurança da informação, para as quais há material de pesquisa que remonta à década de 1960. Dentre o grande volume de publicações que guardam uma relação com o tema do presente trabalho foram selecionadas quase duas centenas, distribuídos conforme a chave utilizada para a pesquisa, como segue:

- *Abuse & Misuse Case*, 16 documentos;
- *Attack modeling*, 19 documentos;
- *Information Security & Software*, 59 documentos;
- *Malware anatomy*, 2 documentos;
- *Mobile Software Specification*, 17 documentos;
- *Operating System Security*, 5 documentos;
- *Risk Analysis & Software Design*, 9 documentos;
- *Security-Driven Development*, 4 arquivos;
- *Software Design & Information Security*, 13 documentos;
- *Software Testing and Information Security*, 15 documentos;
- *Threat Modeling*, 6 documentos;

Além destes foram acessados mais vinte e cinco documentos de anais do Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais – SBSEG, do período

de 2006 a 2013, incluindo artigos e minicursos abordando o desenvolvimento de software para dispositivos móveis e sistema operacional Android.

1.3 Estrutura do documento

Este trabalho está organizado em cinco capítulos, incluindo o presente, capítulo um, que faz a introdução ao trabalho. No capítulo dois são tratadas as questões relativas às ameaças à segurança da informação em ambientes de computação móvel.

O capítulo três destaca aspectos relevantes para a segurança da informação no ciclo de vida do desenvolvimento de software. O capítulo quatro apresenta o ponto central deste trabalho, a abordagem de desenvolvimento de software dirigido pela segurança e a referência ao estudo de caso com a aplicação do modelo no desenvolvimento de uma aplicação. O capítulo seis apresenta as conclusões apuradas na execução do trabalho, bem como aponta para a continuidade do trabalho em atividades futuras.

Nos apêndices são apresentadas as informações complementares, modelos e exemplos de documentos utilizados e produzidos no decorrer do trabalho.

Capítulo 2 – Ameaças em ambientes móveis

Este capítulo visa contextualizar os aspectos de segurança da informação inerentes aos diversos ambientes computacionais, apresentar a estrutura dos dispositivos móveis, as principais ameaças e vulnerabilidades, os tipos de ataques e as defesas normalmente empregadas. Também são apresentados modelos de identificação de ameaças e ataques presentes na literatura, com uma proposição para a classificação das ameaças baseada nestes modelos.

2.1 Conceitos de Segurança da Informação

A palavra "segurança" abrange muitos significados distintos e por vezes até conflitantes. Conforme (Avizienis et al, 2004), pode relacionar-se a ameaças intencionais, como intrusões, ataques e roubo de informações – designada como *security* na língua Inglesa. Também pode indicar sistemas confiáveis, construídos para tolerar erros de software, de hardware ou dos usuários – do Inglês *reliability*. Ou ainda pode referir-se a problemas causados pelo sistema aos seus usuários ou ao ambiente, como erros de programação que possam até mesmo provocar acidentes – traduzida do Inglês *safety*. Nesta dissertação aborda-se a segurança somente no tocante ao primeiro significado, ou seja, a segurança relacionada à identificação, prevenção e combate às ameaças à segurança da informação, especialmente no que tange ao processo de desenvolvimento de software.

O trabalho de (Avizienis et al, 2004) aborda a segurança da informação sob a ótica da dependabilidade, isto é, a capacidade de um software em fornecer um serviço que pode justificadamente ser confiável, ou a habilidade de evitar, de forma aceitável pelo usuário, falhas no provimento dos serviços, seja em frequência ou em severidade. Para estes autores a segurança é entendida como um conjunto das seguintes características:

- **Confidencialidade**, ou ausência de divulgação não autorizada da informação;
- **Integridade**, ou ausência de alterações indevidas;
- **Disponibilidade** dos serviços corretos para os usuários autorizados;
- **Confiabilidade**, isto é, a continuidade na execução correta dos serviços;
- **Segurança do usuário**, ou seja, a ausência de consequências catastróficas para o usuário e para o ambiente;

- **Manutenibilidade**, quer seja, a capacidade de submeter-se a modificações e reparos;

No estudo, (Avizienis et al, 2004) ressaltam que a segurança da informação tem por objetivo garantir estas características, evitando os incidentes de segurança, ou seja, eventos que atentem contra qualquer uma delas, inviabilizando o uso adequado da informação. Estes incidentes - ou ataques - são eventos provocados por incompetência, descuido, mau uso ou uso de má fé que, explorando a existência de falhas, situações não previstas ou fraquezas do projeto ou em decorrência delas – as vulnerabilidades – provocam o uso impróprio do sistema ou das informações tratadas pelo mesmo. Seus autores observam que, para reduzir ou mitigar essa possibilidade, são adotadas as defesas ou contramedidas de segurança, que são procedimentos, técnicas e ferramentas cujo objetivo é reduzir o risco dos incidentes, seja pela diminuição da probabilidade de um ataque por meio da eliminação das vulnerabilidades, seja pela adoção de defesas ou contramedidas que possam reduzir a perda em potencial representada por um possível ataque, já que o risco é o produto da probabilidade do ataque pela perda em potencial que o mesmo pode causar.

Ribeiro (2002) caracteriza a informação, em seu estudo voltado para a segurança da informação, como um bem, um ativo muitas vezes intangível, mas geralmente de grande valor. Na atual era da informação, o volume de informação que se produz, se manipula e se armazena é muito elevado, dada a facilidade de fazê-lo por meios eletrônicos. Embora a informação possa manifestar-se em diversos meios – impressa ou eletrônica, analógica ou digital, etc. - é no modelo digital e eletrônico que tem seu expoente em termos de volume, flexibilidade e facilidade de uso e acesso. Nesse contexto, a segurança da informação vem a ser a proteção dessa informação contra as ameaças que comprometam suas qualidades fundamentais, tornando-a inútil.

Ainda de acordo com Ribeiro (2002), a segurança da informação é algo tão esperado pelos usuários quanto o desempenho ou o uso otimizado dos recursos computacionais, mesmo que não faça parte das especificações iniciais do sistema. O usuário de qualquer sistema espera que esse seja intrinsecamente seguro a despeito de ter solicitado ou não com esta característica. E caso não receba o que espera, certamente não ficará satisfeito. Para o autor a segurança da informação contempla também:

- A necessidade de **autenticação**, que garante ao usuário ou ao sistema que seu interlocutor é realmente quem diz ser.

- O **não repúdio**, que é a capacidade de comprovar quem executou determinada ação.
- A **legalidade**, que trata da **conformidade** do uso da informação com os dispositivos normativos, regulatórios e legais.
- A **privacidade**, que tem o propósito de restringir, além do acesso indevido, a vinculação de um usuário a uma atividade executada.
- A **auditabilidade**, cujo objetivo é o rastreamento e a reconstituição de toda a movimentação e das alterações da informação em função do tempo.

Ribeiro (2002) também salienta que, embora a segurança da informação não seja dependente apenas do sistema, pois além da dependência do próprio hardware e da infraestrutura que o sustenta - incluindo-se a de comunicação, as redes - são necessários diversos controles físicos, normas e procedimentos para garanti-la, é no software que se originam a maioria dos problemas de segurança da informação. Assim, com o uso cada vez maior de sistemas conectado via Internet, com a utilização para as atividades profissionais, com o crescimento da capacidade de processamento e armazenamento e com a diversidade dos dispositivos de computação para as mais diversas atividades humanas, a preocupação com a segurança da informação tornou-se prioritária, e as respostas fornecidas frente aos problemas são, por ora, inferiores aos desafios apresentados (Zhou; Jiang, 2013).

2.2 Ameaças, Falhas, Erros, Faltas e Vulnerabilidades

Embora o esperado seja a confiabilidade total, de modo que as atividades possam ser realizadas por completo e sem interrupções, todo software está exposto a ameaças que atentam contra esta premissa. Estas ameaças estão presentes e se manifestam não somente no software, mas em todos os componentes do ambiente computacional.

Em seu estudo, (Avizienis et al, 2004) considera que o não provimento correto de serviços e da comunicação de dados por um sistema pode ter origem intencional (causada por uma lógica maliciosa ou por intrusões) ou acidental (causada por falhas físicas, de projeto ou decorrente da interação). Segundo (Avizienis et al, 2004) um sistema é uma entidade que interage com outras entidades (outros sistemas, incluindo hardware, software, seres humanos e o mundo físico com seus fenômenos naturais) e que possui capacidade de computação e de comunicação caracterizadas por quatro propriedades fundamentais: funcionalidades, performance, dependabilidade (ou confiabilidade) e custos. Essas propriedades podem sofrer

influências da usabilidade e da adaptabilidade do sistema, isto porque um sistema provê um serviço ao usuário por meio de suas interfaces externas, que podem ser ameaçados por falhas, erros e faltas.

Também segundo (Avizienis et al, 2004), um serviço a ser provido pelo sistema pode falhar por não atender a especificação funcional ou porque essa especificação não reflete corretamente a necessidade, levando o serviço de seu estado correto para um estado incorreto, ou indisponibilidade, para o qual deverá ser providenciada uma recuperação. Esse desvio ou mudança de estado – de correto para incorreto, como mostrado na Fig. 1 – é denominado erro, e a causa do erro é uma falta. Ou seja, um erro é um estado do sistema que pode levar a uma falha no serviço prestado por esse sistema, isto é, uma falha ativa. Esta sequência de eventos é mostrada na Fig. 2. Entretanto, alguns erros podem não ocasionar uma falha de imediato ou na sequência das operações de determinada funcionalidade, redundando em uma falta dormente ou inativa.

As falhas também podem resultar em uma degradação do serviço prestado, com redução do desempenho, inexatidão ou entrega parcial dos serviços, ou seja, em uma falha parcial que não comprometa todo o sistema, fazendo-o operar de forma mais demorada, em regime parcial, limitado ou de emergência. Os meios para atingir a confiabilidade necessária aos sistemas podem ser agrupados, conforme (Avizienis et al, 2004), em:

- **Prevenção de falhas**, isto é, formas de prevenir a ocorrências ou a introdução de falhas nos sistemas;
- **Tolerância a falhas**, ou seja, evitar a falha dos serviços mesmo na ocorrência de faltas nos sistemas;
- **Remoção de falhas**, o que significa reduzir o número e a gravidade das falhas;
- **Previsão de falhas**, que implica em estimar a quantidade atual, a incidência futura e as prováveis consequências das falhas.

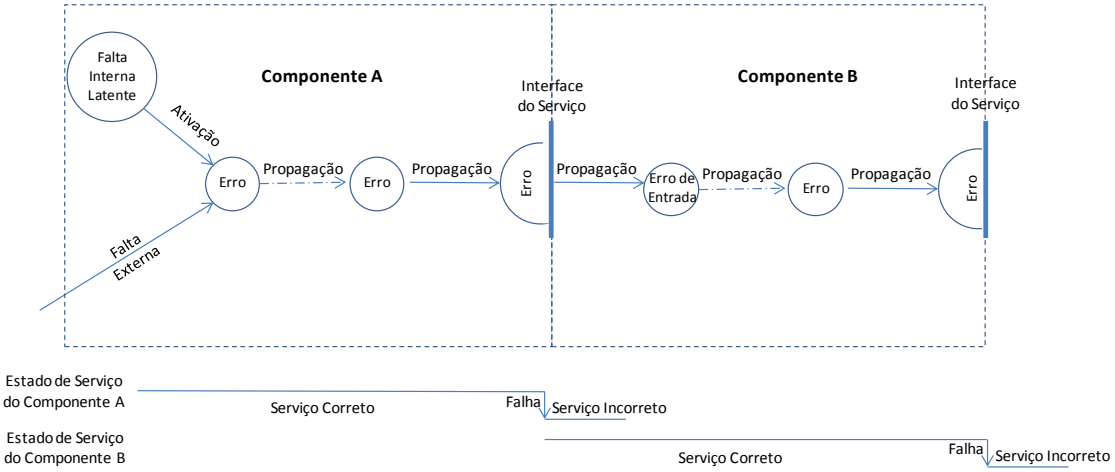


Figura 1 - Propagação do Erro.
Adaptado de (Avizienis et al, 2004).

A prevenção e a tolerância a falhas visam obter a capacidade de oferecer um serviço confiável, enquanto que a remoção e a previsão de falhas têm como propósito atingir a confiança nessa capacidade, assegurando que as especificações funcionais e de confiabilidade são adequadas e que o sistema é capaz de atingi-las.

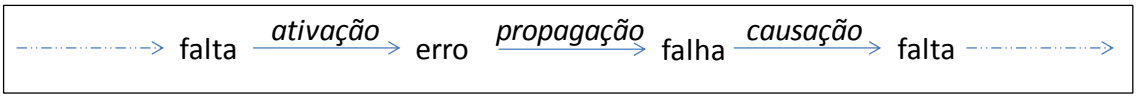


Figura 2 - Cadeia das ameaças.
Adaptado de (Avizienis et al, 2004).

As ameaças estão presentes em todo o ciclo de vida dos sistemas, que pode ser separado em fase de desenvolvimento e fase de uso (Avizienis et al, 2004). Durante a fase de desenvolvimento o sistema está exposto ao ambiente de desenvolvimento, no qual falhas de desenvolvimento podem ser introduzidas no sistema pelos elementos desse ambiente, quer sejam, o mundo físico, os desenvolvedores humanos, as ferramentas de desenvolvimento e as instalações de produção e testes. A fase de uso inicia-se a partir da aceitação do usuário, e então inicia-se a entrega dos serviços providos pelo sistema. Nessa fase, uma falha de serviço provoca a interrupção dos serviços, porém pode haver também desligamentos intencionais e autorizados para a manutenção do sistema – ressaltando que tal manutenção pode ser corretiva, preventiva, adaptativa e evolutiva, como mostra o Quadro 1, e trata-se de um processo de desenvolvimento como tal. Na fase de uso o sistema interage com o ambiente de uso e com os seguintes elementos:

- O **ambiente físico**, com seus fenômenos naturais;
- Os **administradores**, humanos ou outros sistemas, incluindo os responsáveis pela operação e manutenção do sistema, os quais tem autoridade para gerenciar, modificar, reparar e usar o sistema, considerando que tais elementos podem não ter a competência adequada ou até mesmo ter objetivos escusos;
- Os **usuários**, aqueles elementos que vão receber os serviços providos pelo sistema por meio de suas interfaces;
- Os **provedores de serviços** para o sistema e suas interfaces;
- Os **recursos** que provêm serviços especializados ao sistema, seja como fonte de informação – tais como relógios, GPS, rádio – ou como serviços especializados - tais como links de comunicação, baterias, fontes de energia e mecanismos de controle de temperatura.
- Os **intrusos**, que não estão habilitados ou autorizados, porém tentam acessar os serviços do sistema, alterá-los ou interrompê-los, alterar suas funcionalidades ou performance, acessar informações confidenciais. São, por exemplo, os crackers e hackers, os agentes maliciosos ou de governos/instituições (hostis ou não) e os info-terroristas.

Manutenção			
Reparos		Modificações	
Corretiva	Preventiva	Adaptativa	Evolutiva
Remoção das falhas já identificadas	Busca e remoção de falhas latentes	Ajustes às mudanças ambientas	Expansão das funções

Quadro 1 - Os tipos de manutenção
Adaptado de (Avizienis et al, 2004)

De acordo com (Avizienis et al, 2004) as falhas afetam o sistema durante todo o seu ciclo de vida, e podem ser classificadas em oito classes elementares:

1. Por fase de criação ou ocorrência, que podem ser de desenvolvimento ou de uso;
2. Por região de ocorrência, que podem ser internas ao sistema ou externas a ele, porém provocando ou propagando erros;
3. Por fenômeno, que podem ser naturais ou provocados por seres humanos;
4. Por dimensão, sendo as que se originam ou afetam o hardware ou as que se originam de ou afetam o software;
5. Por objetivo, que pode ser malicioso ou não;
6. Por intenção, que pode ser deliberada ou não;
7. De capacitação, que se dividem em acidentais ou por incompetência;
8. Por persistência, que podem ser permanentes ou temporárias;

As falhas por fenômeno humano e de caráter malicioso – geralmente causadas por *malware* ou tentativas de intrusão - são as que mais têm afetado os sistemas para dispositivos móveis. As falhas acidentais ou por incompetência também são um conjunto expressivo, e o crescente uso de COTS - *Commercial Off-the-Shelf* - componentes de prateleira, acaba aumentando o risco de ocorrência de tais falhas.

As falhas de interação – ou operacionais – ocorrem durante a fase de uso do sistema, e são decorrentes da interação do sistema com o ambiente e com os usuários. Podem ocorrer em função de problemas na configuração ou reconfiguração dos parâmetros operacionais, instalação, manutenção ou atualização do sistema. Geralmente uma falha desse tipo ocorre devido a uma vulnerabilidade, isto é, uma falha interna que possibilita a uma falha externa – geralmente de causa maliciosa – atingir o sistema.

As vulnerabilidades podem ser de desenvolvimento ou falhas operacionais. Elas podem ser maliciosas ou não mal-intencionadas, tal como pode ser a falha externa que as exploram. Nesse aspecto, nota-se que há semelhanças interessantes e óbvias entre uma tentativa de intrusão que explora uma vulnerabilidade (os *exploits*) e uma falha externa física. A vulnerabilidade também pode ser resultado de uma falha de desenvolvimento deliberado, por questões de redução de custo ou por questão de usabilidade, resultando em uma proteção limitada, ou mesmo na ausência total de proteção.

As falhas físicas estão mais diretamente relacionadas a aspectos ambientais que interferem no *hardware*, tais como interferência eletromagnética, ruídos e problemas da alimentação elétrica ou de temperatura de operação e, de certo modo, podem ser também consideradas como vulnerabilidades.

As falhas decorrentes da especificação funcional resultam em funcionalidades que, embora estejam compatíveis com a sua descrição, não atendem as necessidades do usuário. São decorrentes de má interpretação, premissas incorretas, inconsistências ou mesmo erros de transcrição. O problema com este tipo de falha é que somente pode ser detectada em função de suas consequências, isto é, após a ocorrência. Geralmente geram um resultado diverso do esperado, seja em função do conteúdo apresentado ou em função do tempo necessário para a entrega do resultado - falha de performance.

Em todo caso, essas falhas podem resultar na interrupção do serviço ou na execução do serviço de forma irregular. Do ponto de vista da identificação, tais falhas podem ser claramente identificáveis ou não, pois podem resultar em uma degradação do serviço tolerável pelo usuário.

Além disso, tais falhas podem ser percebidas de maneira igual por todos usuários, ou usuários distintos podem ter percepções distintas dessas falhas.

O estudo de (Avizienis et al, 2004) reforça que as consequências das falhas também podem ser utilizadas para uma classificação mais acurada, em função das características da segurança, como por exemplo:

- Para a confidencialidade, o tipo de informação que pode ser indevidamente divulgada;
- Para a integridade, a extensão da corrupção dos dados e a capacidade de recuperá-los;
- Para a disponibilidade, a duração da interrupção dos serviços;
- Para a segurança, a possibilidade de ameaças a vidas humanas.

Tais consequências podem variar também na magnitude dos danos – consequentemente dos custos - causados pela falha, podendo variar de imperceptível ou aceitável até proporções catastróficas. Um serviço interrompido ou finalizado em decorrência de falha irá requerer uma restauração, que pode ser manual ou automática, implicar em uma recuperação, reinicialização ou *reboot* do sistema todo. Pode também necessitar de uma manutenção corretiva, isto é, de um novo desenvolvimento, atualização ou troca de componentes.

As falhas típicas de desenvolvimento geralmente ocorrem em função de problemas de orçamento ou em função do cronograma, resultando na redução de atividades do processo de desenvolvimento - invariavelmente as de testes – e na entrega de sistemas tecnologicamente frágeis ou funcionalmente inadequados às necessidades do usuário. Além disso, contribuem para este tipo de falha:

- Sucessivas e recorrentes alterações de funcionalidades, exigindo uma constante verificação de possíveis vulnerabilidades e resultando na introdução de novas falhas a cada alteração;
- Projeto inadequado, produzindo um desempenho ou funcionalidades aquém do esperado;
- Uma grande quantidade de erros, que pode superar a capacidade de resolução em tempo de desenvolvimento;
- Baixo nível de confiabilidade nas avaliações e simulações, indicando que o sistema não atingirá o nível desejado de confiabilidade;
- Falha na orçamentação, seja em custos, seja em prazo ou ambos, subestimando a complexidade do sistema.

Tais problemas podem resultar em estouro do orçamento ou dos prazos, ou na entrega de sistemas incompletos, com funcionalidades, desempenho ou confiabilidade inferiores às necessárias ou especificadas. Convém ressaltar aqui que geralmente a confiabilidade, a segurança e os custos são tratados em especificações não-funcionais dos sistemas, quando, de fato, devem ser meta-especificações, dado o impacto de tais requisitos sobre o resultado dos serviços propiciados pelos sistemas (Avizienis et al, 2004).

2.3 Dispositivos móveis (*mobile*)

Os dispositivos de computação móvel - em especial os *tablets* e os *smartphones*, têm experimentado um grande crescimento tanto em número de equipamentos, como mostrado na Fig. 3, quanto no de aplicações, mas também em ataques que comprometem a segurança da informação de seus usuários (La Polla et al, 2013). A expansão dos recursos de comunicação à disposição destes dispositivos, tais como redes 3G, 4G e 5G, Wi-Fi, *Bluetooth* e NFC, aliadas à aplicação cada vez mais intensa em processos de negócio das empresas ou em atividades pessoais que tratam informações de valor, vem tornando esse ambiente um alvo muito visado pelos atacantes.

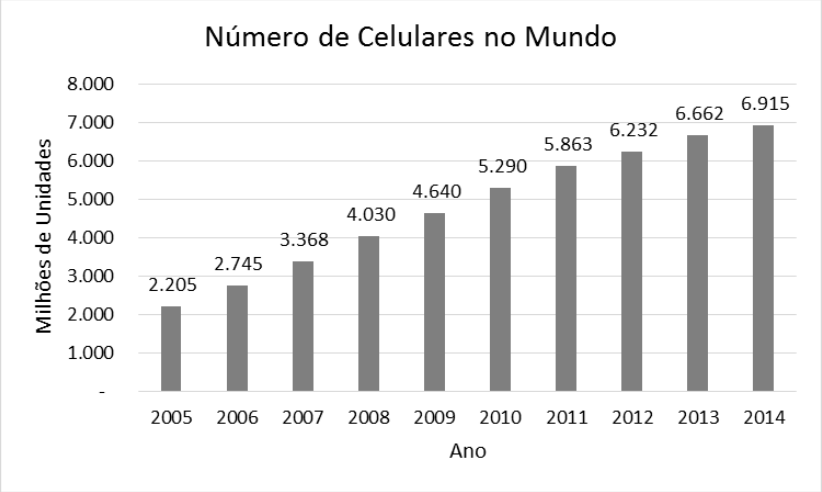


Figura 3 – Crescimento do número de celulares no mundo.
Adaptado de (UIT, 2014)

Este crescimento é notado em todos os segmentos do uso dos dispositivos, bem como nos diversos sistemas operacionais, conforme mostrado na Tabela 1, mas acentua-se no caso do Android, sistema operacional distribuído pela Google, uma vez que este sistema operacional

equipa mais que dois terços dos dispositivos comercializados nos últimos cinco anos (Gartner, 2014).

Tabela 1 - Vendas de Equipamentos por Sistema Operacional.
Adaptado de (Gartner, 2014).

Sistema Operacional	3T14		3T13	
	Unidades X 1000	% Mercado	Unidades X 1000	% Mercado
Android	250.060	83,07%	250.243	84,74%
iOS	38.186	12,69%	30.330	10,27%
Windows	9.033	3,00%	8.916	3,02%
BlackBerry	2.419	0,80%	4.401	1,49%
Outros	1.310	0,44%	1.407	0,48%
TOTAL	**301.008**	**100%**	**295.297**	**100%**

Comparado ao número de equipamentos, apontado pela UIT (2014) em cerca de sete bilhões de aparelhos ao final de 2014, e ao número de usuários, a quantidade de *malwares* não é significativa, porém vem crescendo de forma a comprometer os avanços obtidos em termos de aplicação destes dispositivos. Boa parte da infraestrutura tecnológica empregada pelos dispositivos móveis não foi inicialmente projetada para suportar tamanho crescimento nem tampouco o intenso uso para o tráfego de dados - como no caso das opções disponíveis para a conectividade, tais como GSM, GRPS/EDGE, UMTS, HSPA, *Bluetooth*, NFC e até o próprio IEEE 802.11. Tampouco os sistemas operacionais e as aplicações destes dispositivos, limitados pela capacidade de processamento, armazenamento e disponibilidade de serviços.

Desta forma, o cenário atual requer uma criteriosa avaliação com o intuito de prover condições para os usuários destes equipamentos e para os desenvolvedores de software que lhes permitam fazer uso de forma segura e efetiva de seus recursos, seja para o lazer ou para uso profissional, para acesso online a bancos ou compras na internet.

De acordo com o estudo de (La Polla et al, 2013), em um ambiente de dispositivos móveis diversos atores contribuem para a mitigação dos riscos relativos à segurança da informação. Os fabricantes de hardware, que às vezes fornecem também o sistema operacional e bibliotecas para desenvolvimento de aplicações, respondem pela segurança física, mecânica e computacional no que diz respeito à confiabilidade do equipamento e a compatibilidade com funcionalidades, normas e regulamentos. As companhias operadoras provêm a infraestrutura de comunicação e também ofertam facilidades e serviços distintos, que dependem das funcionalidades do hardware e do sistema operacional e também estão sujeitos a normas e

regulamentos bastante diversificados em função da atuação geográfica e da legislação. Os desenvolvedores de software utilizam as funcionalidades e facilidades para prover as aplicações, as quais são exploradas pelos usuários.

Cada um desses atores tem responsabilidades sobre a segurança da informação nesses ambientes, e segundo (La Polla et al, 2013) podem ser alcançados com abordagens distintas quanto à prevenção a ataques e contramedidas de segurança, a saber:

- Usuários devem ser informados dos possíveis problemas, orientados e educados a utilizarem seus dispositivos corretamente e de modo seguro;
- Desenvolvedores devem adotar medidas de proteção à segurança da informação, as mais atualizadas possíveis e que permeiem todo o ciclo de vida do software, tendo como referência as vulnerabilidades e os ataques já conhecidos;
- Operadoras devem reforçar os mecanismos de identificação e defesa das redes, e adotar medidas preventivas contra os ataques;
- Fabricantes devem ser ágeis na identificação de falhas e na atualização do hardware, sistemas operacionais e bibliotecas com o intuito de reduzir ou eliminar falhas e vulnerabilidades que possam motivar ataques.

Como mostrado na Fig. 4, a hierarquia formada por esses atores tem na sua maior expressão os usuários, já que o número desses é muito maior do que a quantidade de desenvolvedores, que superam em muito o número de operadoras, o qual é bem superior ao número de fabricantes.

É notório que as falhas mais graves ocorrem no domínio dos desenvolvedores e fabricantes das plataformas. Porém, no que diz respeito à segurança da informação em geral – e particularmente nos ambientes dos dispositivos móveis – o histórico da maioria das falhas – e consequentemente dos ataques – revela o uso impróprio e a problemas do software. Esta é a razão pela qual este trabalho propõe a atuação prioritária nesses componentes e junto a esses dois atores: usuários e desenvolvedores.

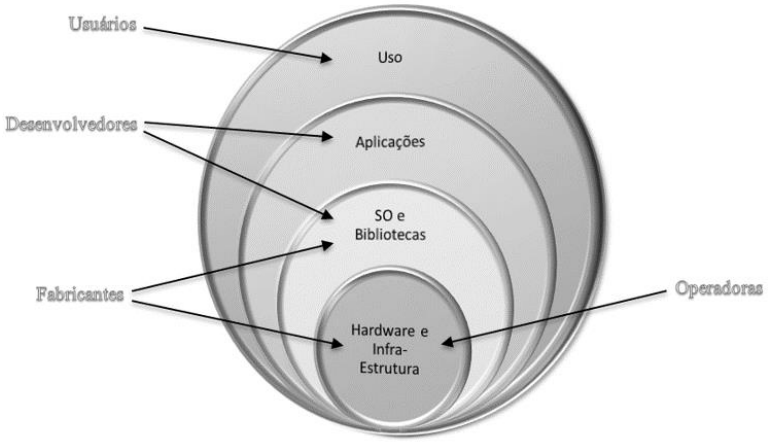


Figura 4 - A hierarquia da segurança da informação nos dispositivos móveis.
Adaptado de (La Polla et al, 2013).

2.3.1 A infraestrutura

A tecnologia disponível para os dispositivos móveis conectarem-se às redes divide-se basicamente em duas modalidades: a rede de telefonia e as redes sem fio. No primeiro modo estão as tecnologias criadas para a mobilidade – porém não necessariamente para o tráfego de dados – como GSM, GPRS, EDGE e UMTS. A principal característica dessa tecnologia é o uso para a comunicação por voz e mensagens curtas de texto (SMS), que, aprimorada, possibilitou a expansão do uso para mensagens multimídia (MMS), aplicações WAP, e-mail, teleconferências e finalmente acesso à Internet (La Polla et al, 2013).

No segundo modo estão as tecnologias de rede local sem fio – WLAN, entre as quais destacam-se o *Bluetooth* e o Wi-Fi (IEEE 802.11). Apesar de terem sua origem na necessidade de conexão de equipamentos móveis – como os notebooks - às redes locais, essas tecnologias não privilegiam a mobilidade, mas sim a conectividade com redes de tráfego de dados, alta velocidade e curta distância, no intervalo de uma dezena a poucas centenas de metros (La Polla et al, 2013).

2.3.2 Particularidades dos dispositivos móveis

Ao tratar de segurança da informação em ambiente de dispositivos móveis há que se considerar que existem diferenças importantes entre este ambiente e o ambiente tradicional de computação pessoal (PC) – notebooks e desktops. Estas diferenças são primordiais na avaliação dos riscos e na disposição de soluções para os problemas de segurança da informação de tais dispositivos, pois ao mesmo tempo em que ampliam as vulnerabilidades – e, portanto, as opções de ataque –

reduzem o espaço e os recursos para a utilização de mecanismos tradicionais de defesa. Dentre as principais diferenças, cabe citar as elencadas por (La Polla et al, 2013):

- Mobilidade: Os dispositivos móveis estão disponíveis para uso e acesso de seus usuários praticamente o tempo todo, e não somente nas residências ou ambiente de trabalho, mas também em locais de acesso público e coletivo, como meios de transporte, shoppings, instituições e órgãos públicos, entre outros. Evidentemente essa exposição constante, aliada à falta de mecanismos de alerta e proteção adequados, favorece os ataques e a proliferação de problemas de segurança da informação.
- Personalização: Por tratarem-se de dispositivos que foram incorporados ao conjunto de acessórios tecnológicos de uso pessoal, mais especificamente por estarem associados a um número de telefone, os dispositivos móveis são submetidos a um elevado índice de personalização, exigindo uma abordagem diferenciada no que diz respeito à identificação de vulnerabilidades e hábitos de uso. Este aspecto é particularmente significativo quando se trata da abordagem comportamental para a identificação de falhas e problemas, ou do estabelecimento de um padrão de comportamento para a identificação de tentativas de ataques ou fraudes.
- Conectividade: Os dispositivos móveis fazem uso intenso da conectividade proporcionada pela infraestrutura de comunicação para dispositivos móveis, seja pela rede de telefonia móvel celular, redes Wi-Fi, Wi-Max, Bluetooth e NFC. Além disso, tais dispositivos podem permanecer conectados à mais de uma rede simultaneamente, funcionando como gateway, bridge ou ponto de acesso.
- Convergência de tecnologias: Para prover os diversos serviços oferecidos aos usuários destes dispositivos, inúmeras tecnologias são empregadas, partindo da comutação de circuitos de voz até a utilização de pacotes de dados, passando por sistemas de mensagens curtas (SMS) e multimídia (MMS), streaming de áudio e vídeo, recepção de TV digital e outras. Todas essas tecnologias são gerenciadas por um único mecanismo de hardware e pelo mesmo sistema operacional, compartilhando capacidade de processamento e armazenamento e as interfaces do dispositivo.
- Capacidades reduzidas: Em que pese o contínuo avanço impulsionado pela crescente demanda de recursos computacionais, os dispositivos móveis ainda

dispõem de menos recursos quando comparados à computação pessoal - os PCs. A capacidade de processamento, as interfaces restritivas e a capacidade de armazenamento são reduzidas e restringem o uso de mecanismos de defesa mais aprimorados, como antivírus ou *firewalls*, comuns na plataforma PC. Outro aspecto crucial e que implica em restrição é a capacidade das baterias que alimentam o dispositivo, que são bastante demandadas pelos circuitos de recepção e transmissão de sinal de rádio – RF – e precisam ser poupadas com a redução do uso do processador e dos acessos à memória.

2.4 Sistemas operacionais de dispositivos móveis

A segurança da informação entendida em um aspecto amplo, no qual impõe-se como condição a proteção de todos os recursos computacionais voltados para o provimento de serviços e, portanto, de informação, passa necessariamente pela segurança do sistema operacional, um dos principais componentes de praticamente todo sistema computacional.

Os sistemas operacionais para ambiente de dispositivos móveis são especialmente importantes para a segurança da informação, uma vez que são profundamente adaptados aos recursos computacionais e à infraestrutura de serviços e funcionalidades específicas do ambiente e dos equipamentos. São vastos os estudos sobre os principais sistemas operacionais para dispositivos móveis utilizados pelos maiores fabricantes destes equipamentos, tais como o de (Hammershøj et al, 2010), que aborda as diversas particularidades com foco nos aspectos de segurança.

Como o foco deste trabalho é o desenvolvimento de aplicações para dispositivos móveis, e neste ambiente destaca-se o sistema operacional Android, são apresentadas e avaliadas as principais características deste SO, que é baseado no núcleo do Linux, é de código aberto e com uma arquitetura voltada a serviços, como mostrado na Fig. 5. Por estas características a evolução do Android tem apresentado significativas melhorias no que tange à segurança da informação, e as respostas às falhas e ameaças são mais eficazes e mais rápidas do que as de seus concorrentes (Braga et al, 2012).

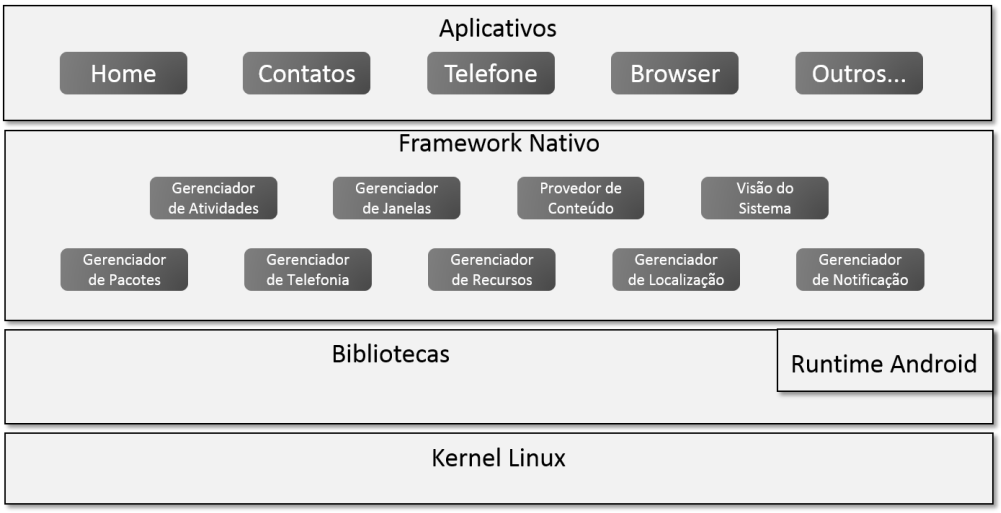


Figura 5 - Arquitetura do Android.
Adaptado de (BRAGA et al, 2012).

No trabalho de (Shabtai et al, 2010) os pesquisadores apresentam um abrangente estudo do Android com o foco em segurança, contemplando os mecanismos de segurança do próprio sistema, uma avaliação dos aspectos de segurança derivados das aplicações e dos recursos e também contramedidas - propostas de soluções e tratamento para ameaças. Quanto às vulnerabilidades e ameaças, Zhou e Jiang (2013) avaliaram o sistema operacional durante quase dois anos, identificando características e comportamento do ambiente e dos ataques, o que permitiu a criação de uma categorização das ameaças e um processo para identificar ameaças em potencial, antevendo o processo de evolução das ameaças e a consequente necessidade da evolução das respostas e contramedidas à estas ameaças. (Braga et al, 2012) apresentam um detalhado estudo do funcionamento do Android, vulnerabilidades e ameaças, além de orientações quanto a cuidados no desenvolvimento de software para a plataforma, ferramentas e mecanismos de defesa.

Já o estudo de Gold (2012) avalia os aspectos de construção do Android, sua evolução histórica e às respostas as ameaças, fazendo um prognóstico quanto aos desafios a serem enfrentados pelos desenvolvedores para dar garantia aos usuários do Android. Mesmo sendo exaustivamente pesquisado, algumas questões cruciais ainda permanecem carentes de respostas confiáveis, como por exemplo a autenticação do usuário, as permissões requeridas pelas aplicações e o tratamento de informações compartilhadas entre aplicações multitarefas.

O **Apple iOS**, também chamado **iPhone OS**, em parte devido ao fato de ser proprietário e voltado apenas para os equipamentos produzidos pela Apple, apresenta uma maior robustez

e não é tão explorado quanto o Android. Porém é crescente o surgimento de *malwares* para esse ambiente, bem como as iniciativas da própria Apple para fazer frente a essas questões. Um exemplo desse esforço é que a arquitetura do iOS, mostrada na Fig. 6, provê APIs de segurança na camada Core Services e a evolução dos *security services* na versão iOS 7 (Tracy, 2012).

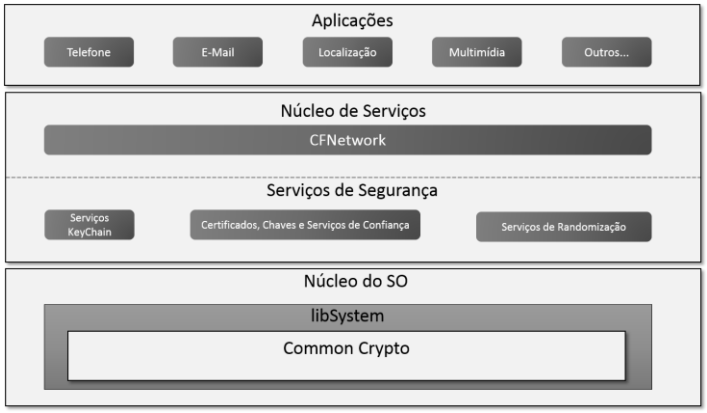


Figura 6 – Arquitetura do Apple iOS.
Adaptado de Tracy (2012).

O **Windows Phone**, sistema operacional da Microsoft para dispositivos móveis, é o sucessor do Windows CE e do Windows Mobile e faz parte da unificação promovida pela Microsoft quanto à arquitetura e a apresentação dos seus sistemas operacionais. A necessidade de controle das aplicações desenvolvidas por terceiros é uma iniciativa que tem por objetivo a segurança da informação, uma vez que exige a homologação das aplicações antes da distribuição pela Microsoft. Os mecanismos de segurança incluem a integridade do sistema, o acesso seguro, a segurança das aplicações e a proteção dos dados, cada um composto de um conjunto de funcionalidades oferecidas aos usuários e aos desenvolvedores de aplicações (Adibi, 2014). A integração total com os produtos do pacote Office – Office 365 – e o uso de serviços Microsoft na nuvem também reforçam os objetivos de segurança, como proposta para prover a segurança por meio da segregação.

O sistema operacional **Black Berry OS** da empresa Canadense RIM – Research-In-Motion - é uma plataforma proprietária que suporta aplicações J2ME por meio de uma máquina virtual Java disponível no *firmware* dos dispositivos, o que o torna bastante resistente às ameaças. Assim o sistema operacional não é compilado para código de máquina, e provê abstrações para as funcionalidades do hardware dos dispositivos, reduzindo os gargalos de acesso ao hardware. Graças a questões como estas o Black Berry é bastante utilizado por agentes de negócios que demandam alta segurança para a troca de mensagens e e-mails, entre

outras. O fabricante detém o controle sobre a qualidade das aplicações e responde totalmente pelo desenvolvimento do sistema operacional e aplicativos (Mylonas et al, 2011), embora nas versões mais recentes tenha sido implementada uma compatibilidade com aplicações para o Android.

Desenvolvido orginalmente pela Symbian Ltd. e posteriormente incorporado pelo consórcio entre as fabricantes Nokia, Sony Ericsson e a operadora NTT DoCoMo, o sistema operacional **Symbian** é atualmente suportado pela Accenture, e voltado para os equipamentos da Nokia Networks (Nokia Siemens) e da Sony-Ericsson. Sua arquitetura específica – mostrada na Fig. 7 - dificultou a existência de ataques no princípio de seu uso, porém o fato de ser de código parcialmente aberto – algumas interfaces são proprietárias – e de ter sido o primeiro SO para sistemas de dispositivos móveis com interface gráfica logo chamou a atenção, tendo sido vítima do primeiro ataque por um *worm* – o Cabir - ocorrido em 2004. A segurança do sistema é baseada na administração de capacidades – básicas, estendidas e fabricante – que definem as funcionalidades acessadas pelas aplicações (Mylonas et al, 2011).

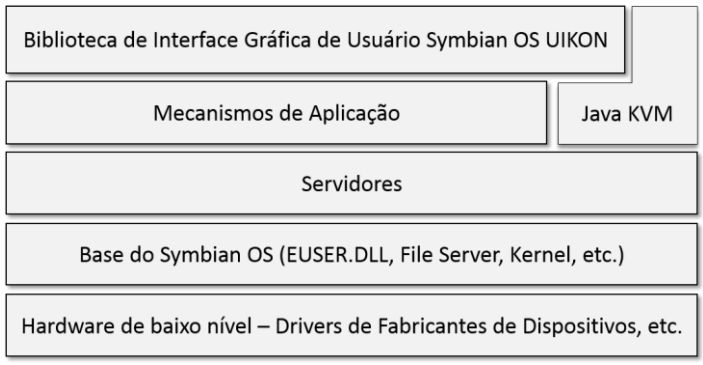


Figura 7 - Arquitetura do SO Symbian.
Adaptado de http://developer.nokia.com/Community/Wiki.

2.5 Ameaças em ambiente de dispositivos móveis

No estudo sobre ameaças e *malwares* de (La Polla et al, 2013) há uma análise dos tipos de ataques aos quais os dispositivos móveis estão sujeitos. Os autores enumeram as características destas ameaças e ataques, classificando-os em função de sua origem ou alvo da seguinte maneira:

- Rede de telefonia e comunicação sem fio: envolve as ameaças que se aproveitam de vulnerabilidades decorrentes do uso da infraestrutura de comunicação da telefonia,

além das capacidades de comunicação por WiFi, *Bluetooth*, NFC, troca de mensagens por SMS e MMS;

- Interrupções: referem-se às vulnerabilidades do sistema operacional exploradas pelas ameaças que interceptam ou sobrepõem-se às chamadas aos serviços disponibilizados às aplicações. Estas vulnerabilidades podem também ser decorrentes das bibliotecas de funcionalidades que compõem o ambiente operacional das aplicações;
- Infraestrutura: nessa categoria encontram-se os ataques que exploram as vulnerabilidades dos recursos computacionais e funcionais do ambiente de dispositivos móveis, tais como câmera, microfone, GPS, acelerômetro, processador, memória, armazenamento, bateria, etc.;
- *Malwares*: contemplam todas as ameaças compostas por *software*, tais como vírus, *worms*, *spywares*, *adware*, *rootkits*, *spam* e outros;
- *Botnets*: além de software propriamente dito, compõe-se de uma diversidade de ambientes computacionais à disposição de atacantes que os controlam para a realização de ataques em massa à ambientes menos vulneráveis;
- Usuários: Os próprios usuários podem representar uma ameaça à segurança da informação, na medida em que, por desconhecimento, descuido ou imprudência, criam condições para a perda da confiabilidade ou estabelecimento de uma vulnerabilidade. Comportamentos como não usar criptografia ou senhas, ou usar senhas fracas, compartilhamento de senhas ou dispositivos, não atualização de *software*, procedimentos repetitivos e previsíveis, e a inobservância das recomendações dos fornecedores são, geralmente, o princípio de problemas de consequências significativas.

2.5.1 *Malwares*

Os *malwares* – do Inglês *Malicious Software*, são trechos de código ou *softwares* projetados para utilizar recursos computacionais de um dispositivo sem o conhecimento ou o consentimento de seu proprietário ou usuário habilitado (La Polla et al, 2013). O propósito desse uso geralmente é ilegal ou desonesto, resultando invariavelmente em prejuízo.

Eles exploram falhas ou fraquezas – as vulnerabilidades – da infraestrutura de rede, dos equipamentos, dos sistemas operacionais, das aplicações ou dos próprios usuários como forma

de propagarem-se e ganharem acesso aos recursos. No estudo citado os *malwares* são categorizados em função de suas características - como a forma de propagação, por exemplo, apresentando a seguinte diferenciação:

- **Vírus**, uma sequência de código cuja finalidade é reproduzir-se em áreas importantes dos dispositivos de armazenamento (discos, *pendrives*, *memory cards*, etc.) ou anexando-se a programas e arquivos.
- ***Worms*** (**vermes**), programas que se propagam por meio de cópias de si próprios para os dispositivos de armazenamento por meio das redes, sem, a princípio, contaminar programas e arquivos;
- ***Trojans***, (cavalos de Tróia), uma forma de software com alguma funcionalidade específica e interessante, como por exemplo, a recuperação de senha de programas ou arquivos protegidos, a conversão de formatos de arquivos de dados ou a geração de números de licença para software licenciado, e que traz em seu código funcionalidades maliciosas com o intuito de explorar as vulnerabilidades;
- ***Rootkits***, códigos maliciosos que normalmente infectam o sistema operacional com o intuito de ocultar operações que demonstram a contaminação - por vírus ou cavalos de Tróia, por exemplo, desabilitar ou superar uma contramedida ou defesa – como antivírus ou firewalls, e também permitir que um usuário mal-intencionado tenha acesso ou mesmo controle o dispositivo infectado;
- ***Botnets***, (de ***Robot Network***), agentes de software autônomos e automáticos cuja finalidade é colocar os dispositivos contaminados a serviço de um controlador, formando assim uma rede de zumbis prontos para executar tarefas como o envio de *spam* ou ataques de DoS;
- ***Spywares***, programas que recolhem informações fornecidas pelo usuário e sobre o uso que este faz do seu equipamento e as transmite a uma entidade externa, geralmente na Internet. Estas informações são posteriormente usadas para burlar sistemas de autenticação e verificação de identidade ou como base para trabalhos de engenharia social com o intuito de possibilitar operações fraudulentas baseadas em falsidade ideológica;
- ***Exploits***, programas ou trechos de código que buscam explorar vulnerabilidades ou falhas de ambientes computacionais – geralmente dos sistemas operacionais – recentemente descobertas para conseguir acesso privilegiado aos sistemas;

- ***Risktools*** ou ***Riskware***, programas ou funcionalidades de programas cujo intuito é avaliar vulnerabilidades do ambiente computacional para então comunicá-las à sua fonte, possibilitando assim a exploração dessas vulnerabilidades de forma mais efetiva. Seu modo de atuação assemelha-se aos Trojans, com a diferença que geralmente trazem um apelo à elevação da segurança do ambiente, promovendo falsas notificações de ameaças ou falhas como reforço para a sua instalação;
- ***Adwares***, programas ou funcionalidades de programas – normalmente shareware – que apresentam anúncios de versões mais completas ou outros produtos do fabricante, e muitas vezes obtém informações sobre o usuário e o ambiente computacional e as enviam, sem o consentimento ou conhecimento do mesmo, para uma base de dados remota, com o intuito de avaliar o perfil e o possível interesse do usuário.

A Tabela 2 apresenta exemplos de *malware* de dispositivos móveis de acordo com essa classificação. Além do propósito, esses *malwares* diferem entre si em características técnicas, tais como:

- A lógica utilizada para a propagação;
- A lógica de controle;
- A forma de coletar as informações;
- A forma de envio das informações obtidas;
- O uso de serviços do equipamento atacado;
- O uso dos serviços de comunicação e rede.

Tabela 2 - Classificação de Malwares.
Adaptado de (La Polla et al, 2013).

Nome	Ano	Tipo	Método de Infecção	Efeito	OS
Cabir	2004	Vírus	Conexão Bluetooth e autocópia	Varredura de Bluetooth contínua, esgota a bateria	Symbian
					Windows OS
Lasco	2005	Vírus	Um virus que se espalha através de redes Bluetooth	Procura e infecta outros telefones	Symbian
Beselo	2008	Vírus	Através de aplicações falsas de MMS e Bluetooth	Carga de MMS	Symbian
Pmcryptic	2008	Vírus	Contamina cartões de memória	Ligações para números específicos para ganhar crédito	Windows Mobile
FakeFlash.C	2012	Trojan	Aplicação Android	Tenta cobrar uma taxa para instalar o Adobe Flash	Android
Gidix.A	2013	Trojan	Simula um aplicativo gerenciador de configurações do sistema	Carrega os dados sensíveis do dispositivo para um servidor remoto Envia silenciosamente mensagens SMS Monitora as chamadas e mensagens SMS enviadas pelo usuário	Android
Avpass.C	2014	Trojan	Distribuído sob a forma de uma aplicação de nome "Clock"	Rouba informações do dispositivo Tenta desinstalar ou desativar os aplicativos relacionados à segurança instalados no dispositivo	Android
Fakeinst.HB	2013	Trojan	Clone reembalado de um popular jogo de corrida de carros	Inclui uma rotina que exige pagamento para continuar a jogar.	Android
CounterClank.A	2013	Adware	Componente de publicidade usado em vários aplicativos suportados por anúncios	Além de exibir os anúncios, provoca vazamentos de informações do dispositivo para um local remoto	Android
SmsReg.A	2013	Risk Tool	É comercializado sob o nome de "Battery Improve"	Recolhe informações do dispositivo. Envia SMS.	Android
BaseBridge.A	2011	Vírus	É distribuído como um arquivo APK chamado "anserverb_qqgame.apk"	Encerra os processos de determinados aplicativos de segurança Tenta enviar SMS e também acessar a internet.	Android
Loozfon.A	2012	Vírus	Requer a instação intencional pelo usuário do dispositivo.	Posta informações confidenciais (lista de contatos, e-mail, número de telefone) em uma URL	Android

Outro modelo de classificação dos *malwares* é o que avalia o comportamento dessas ameaças, como o utilizado por Chandramohan e Kuan (2012), o qual é apresentado no Quadro 2.

No decorrer deste trabalho foi realizado, com o intuito de caracterizar uma base de conhecimento, um estudo mais aprofundado de alguns *malwares* identificados, conceituando-os, quando possível, de acordo com os parâmetros apresentados. O estudo baseia-se nos trabalhos de Chandramohan e Kuan (2012), voltado para as técnicas de identificação de *malware* em dispositivos móveis e formas de proteção, (Shabtai et al, 2010), que avaliam ameaças de alto risco e as proteções do sistema operacional Android, recomendando algumas contramedidas, e de Zhou e Jiang (2013), um abrangente estudo com mais de mil e duzentos tipos de *malwares* com o intuito de caracterizar e classificar os *malwares* com base em características comuns, além de avaliar a evolução deles no decorre do tempo. Também foram pesquisados levantamentos realizados por fornecedores de ferramentas de proteção, como a F-Secure (2014). Os detalhes deste estudo são apresentados no Apêndice A – Anatomia dos *Malwares*.

Quadro 2 - Comportamento de Malwares dos Dispositivos Móveis.
Adaptado de Chandramohan e Kuan (2012)

Comportamento	Descrição
Oferecimento de novidades e diversão	Inicialmente desenvolvido como diversão ou para exibir o conhecimento técnico do autor, é de menor gravidade, porém pode gerar danos não intencionais.
Venda de informações do usuário	Coleta de forma oculta informações e detalhes de uso do equipamento do usuário, tais como contas de e-mail e redes sociais, localização, aplicações instaladas, histórico de downloads e listas de contatos. Estas informações são carregadas para um repositório remoto e depois vendidas para anunciantes e profissionais de marketing.
Roubo de identidade e credenciais do usuário	Captura a identidade e as credenciais do usuário, como detalhes de contas de sites de compras e de contas bancárias, verificando mensagens de texto, capturando as teclas digitadas, procurando em documentos e arquivos ou através de ataques de *phishing*.
Manipulação de entrega de conteúdo	Faz ligações para números especiais para ganhar com a tarifação, envia mensagens de texto para entregar conteúdos diversos, como suporte técnico, cotações de ações, ou serviços para adultos.
Envio de spam por SMS	Envia lotes de mensagens de texto ou multimídia que geralmente contêm anúncios e links de phishing.
Manipulação dos mecanismos de busca	Melhora o *ranking* de sites nos mecanismos de busca.

2.5.2 Evolução das ameaças e o futuro dos *malwares*

Com a massiva disseminação do uso dos dispositivos móveis, há que se considerar um crescimento no número das ameaças à segurança da informação nestes dispositivos, bem como uma diversificação dos métodos e técnicas de ataques, dada a sofisticação cada vez maior dos dispositivos e a ampliação do uso. (La Polla et al, 2013) expõe, em seu levantamento, que houve uma significativa evolução dos *malwares* na década passada, tanto no aspecto quantitativo quanto no que tange às técnicas, em grande parte devido à evolução dos dispositivos e a diversificação dos meios, funcionalidades e componentes que estes dispositivos passaram a oferecer, como cartões de memória, interface USB, câmeras, Bluetooth, MMS e outros.

Zhou e Jiang (2013) apontam outro indicativo de que o futuro sinaliza um crescimento das ameaças: o crescente uso dos dispositivos móveis para o acesso à internet, operações de compras eletrônicas, acesso a funções financeiras e bancárias e o pagamento eletrônico (por meio de tecnologias como NFC, por exemplo). Essas operações já despertaram o interesse do crime organizado, que investe na criação de *malwares* cada vez mais sofisticados. Com vistas a enfrentar esse movimento, há uma proposta da comunidade voltada para a tecnologia de dispositivos móveis que trata da criação de um mecanismo semelhante ao utilizado para detectar e enfrentar epidemias no campo da saúde pública (Szongott et al, 2012).

2.6 Considerações do Capítulo

Neste capítulo foram abordadas as características da segurança da informação e os aspectos dos ambientes de dispositivos móveis, e também as ameaças e *malwares* típicos destes ambientes, com o intuito de favorecer o entendimento do problema da segurança da informação neste contexto. Apesar da amplitude dos assuntos abordados e da necessidade de atualização constante, o conteúdo apresentado é de fundamental importância para possibilitar a identificação das vulnerabilidades e a classificação das ameaças, aprimorando a análise de riscos e as contramedidas a serem adotadas.

Também é indispensável o conhecimento de características do ambiente dos dispositivos móveis e do sistema operacional, além de técnicas para o registro destes conhecimentos de forma padronizada. Isto permitirá criar e manter uma base de conhecimento sobre o conjunto destes aspectos, reforçando a capacidade de identificar novas ameaças e antecipar-se nas medidas de combate às mesmas durante o processo de desenvolvimento de

software. No próximo capítulo serão abordados aspectos deste processo e a relação destes com a segurança da informação.

Capítulo 3 – SDLC e segurança: aspectos relevantes

Mesmo com a toda a evolução ocorrida desde os primórdios da era da computação, e considerando os principais modelos aplicados ao desenvolvimento de software (os quais são abordados no Apêndice B), nota-se que os cuidados com os aspectos de segurança da informação ainda estão mais presentes nas atividades de codificação e validação do software.

Não é regra geral o tratamento dos requisitos de segurança no início do processo de desenvolvimento. Por isso, diversos estudos, como o de McGraw (2005), propõem diminuir o espaço existente entre o processo de desenvolvimento de software e as atividades típicas da segurança da informação. Isto implica em avaliar os usos incorretos ou maliciosos do software e descrevê-los em casos de uso impróprio[1] – evolução dos casos de uso que trata das condições adversas de uso – para prover contramedidas ou respostas que reforcem a segurança da informação.

Essa abordagem leva em conta que a maioria dos modelos acaba usando uma forma iterativa, empregando as melhores práticas de forma cíclica, facilitando sua utilização nos modelos de desenvolvimento ágil. Nestes modelos de desenvolvimento há a vantagem adicional de que, durante as atividades de revisão de cada nova iteração ou ciclo de desenvolvimento, é possível atualizar, identificar e prover respostas às novas ameaças ou a novos casos de uso impróprio, aumentando o escopo da segurança da informação no software.

Com o uso crescente e o aumento da complexidade dos softwares para dispositivos móveis, aliados à capacidade cada vez maior e diversidade dos recursos disponíveis, a segurança da informação é tema recorrente em trabalhos de pesquisa cujo intuito é avaliar, propor e aprimorar técnicas, ferramentas e métodos de garantia desta segurança da informação. Esta preocupação envolve tanto o desenvolvimento quanto o uso de software para dispositivos móveis.

[1] A literatura em geral refere-se a *misuse* ou *abuse case* para designar os usos impróprios ou inadequados – *misuse case* – do software, ou as tentativas de intrusão por meio da exploração de vulnerabilidades do software – *abuse case*. O presente trabalho considera que ambos atentam contra a segurança da informação e, portanto, representam riscos e apresentam resultados indesejados. A opção da tradução para "caso de uso impróprio" e não "caso de abuso" é explicado mais adiante, ao tratar destes artefatos.

Durante a execução do presente trabalho foram estudados diversos processos de desenvolvimento de software, os quais se referem aos paradigmas de desenvolvimento e aos diversos métodos empregados na construção do software. O intuito, ao avaliar as suas características, foi de [2]ressaltar que o modelo, as técnicas e os métodos empregados no processo de desenvolvimento de software influem decisivamente na segurança da informação: alguns modelos favorecem uma abordagem proativa da segurança, enquanto outros a abordam de maneira mais simplificada ou reativa.

De toda maneira, é importante salientar que, a despeito do modelo em questão, e qualquer que seja o método empregado no processo de desenvolvimento, é possível adotar uma abordagem que privilegie a segurança da informação sem que haja o comprometimento dos demais aspectos.

Neste capítulo são apresentados e analisados alguns trabalhos que abordam a segurança da informação no ciclo de vida dos softwares. O enfoque destes trabalhos inclui o tratamento precoce da segurança da informação no processo de desenvolvimento, a utilização de casos de uso impróprio, a análise de riscos e os testes de software voltados para a segurança da informação, em especial para os ambientes de dispositivos móveis.

3.1 Segurança no processo de desenvolvimento de software

Segundo McGraw (2005), a segurança da informação relativa ao processo de desenvolvimento de software ainda enfrenta dois consideráveis problemas. O primeiro refere-se ao fato de que a segurança é abordada nas fases finais do processo. Concentra-se, de fato, na validação e na verificação de requisitos funcionais, com ênfase na etapa de testes, quando pouco pode ser feito para adequá-la aos níveis adequados sem um grande acréscimo de esforço. O segundo é a distância entre a atuação das equipes de desenvolvimento e as equipes voltadas para a segurança da informação, transparecendo que atuam em frentes diferentes e até opostas do problema.

Atento a estes dois aspectos, McGraw (2005) propõe a abordagem precoce da questão da segurança no processo de desenvolvimento, como mostrado na Fig. 8, estabelecendo alguns pontos chave com base nas melhores práticas de segurança da informação:

1. O emprego de **casos de uso impróprio** na etapa de requisitos, explorando o uso impróprio ou o mal-uso do software. Desta forma o projeto pode valer-se do

[2] No Apêndice B são apresentados os aspectos dos processos estudados neste trabalho.

conhecimento adquirido pelos especialistas de segurança para endereçar ameaças e falhas, e consequentemente prover contramedidas a estas, além de reforçar os requisitos de segurança e os cenários de testes com base nestes casos de uso impróprio. McGraw (2005) sugere que a elaboração dos casos de uso impróprio inclua participação dos desenvolvedores, com conhecimento dos requisitos e da arquitetura do software, e de especialistas em segurança, que tenham domínio sobre histórico de ameaças e ataques, ferramentas e técnicas de proteção e defesa, etc.

2. A inclusão da **análise de riscos** na etapa de análise e projeto do software. Esta análise de riscos envolve uma avaliação do ponto de vista do negócio e também da arquitetura do software. Quanto ao negócio, aspectos de custos diretos, tais como responsabilização por danos, perda de produtividade e retrabalho, ou custos indiretos, decorrentes de danos à reputação ou à imagem da corporação, devem ser considerados. Para esta análise os patrocinadores do projeto e os principais interessados devem ser ouvidos, buscando entender deles o que pensam sobre a segurança, o que esperam do software nesse particular, quais os impactos de uma violação da segurança e o que os preocupa. Já no que se refere à arquitetura deve-se analisar os aspectos técnicos que implicam em riscos, e nesse caso o conhecimento do histórico de vulnerabilidades e ataques é essencial, uma vez que mais da metade dos problemas de segurança advêm da arquitetura do software. As pessoas responsáveis por essa análise devem conhecer profundamente os aspectos técnicos do software, e levar em consideração o ambiente computacional como um todo, incluindo o hardware, *frameworks*, compiladores, componentes, etc.

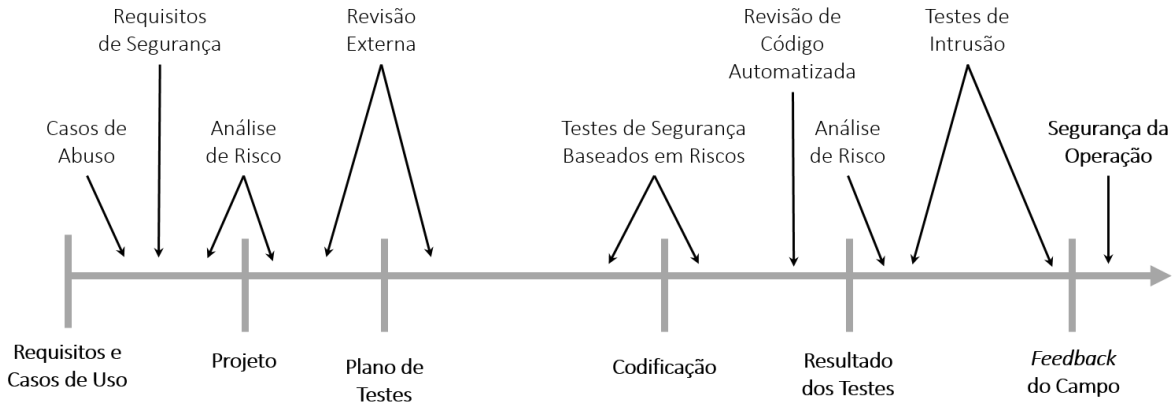


Figura 8 - Abordagem precoce da Segurança.
Adaptado de McGraw (2005).

3. O **planejamento dos testes** contemplando as funcionalidades de segurança e o teste de segurança baseado em riscos. No primeiro caso as especificações das funcionalidades são utilizadas para projetar testes voltados às funções típicas de segurança, tais como a criptografia, a identificação de usuário, o log, a confidencialidade, a autenticação e a autorização, etc. Ou seja, validar o comportamento da segurança do ponto de vista otimista, um teste positivo, buscando aferir a eficácia das funções que visam prover a segurança. Mas como apenas estes testes não são suficientes para garantir a segurança, então deve-se lançar mão dos casos de uso impróprio e dos riscos relacionados para efetuar testes que visam explorar as vulnerabilidades e falhas conhecidas. Este é o segundo caso: um planejamento de teste negativo, que parte do ponto de vista de usuários maliciosos ou do uso impróprio com o intuito de explorar fraquezas que possam comprometer o software. Para McGraw (2005) é importante pensar como um usuário mal-intencionado, um atacante, cujo objetivo é explorar vulnerabilidades ou vencer as defesas do software, e não apenas utilizar os padrões de teste do mercado.
4. A **revisão de código** manual ou automatizada, na etapa de implementação, a qual tem por objetivo identificar falhas ou problemas em potencial decorrentes da atuação dos desenvolvedores ou do uso impróprio de componentes ou interfaces. O uso de ferramentas de análise que ajudam a identificar potenciais problemas de código é essencial nesta etapa, e aqui encontra-se uma dificuldade adicional: não basta conhecer segurança da informação e a ferramenta, mas é necessário que esta revisão seja feita por pessoas ou equipes que tenham experiência também na codificação e no desenvolvimento, o que não é algo comum.
5. A inclusão de **testes de intrusão** na etapa de testes de sistema tem como objetivo explorar os riscos identificados e efetivar as ameaças conhecidas, assim como também comprovar a eficácia das contramedidas adotadas e validar as configurações para a utilização do software. Sendo estes testes embasados na análise de riscos e nos casos de uso impróprio estabelecidos previamente, a objetividade e o alcance dos testes serão maximizados, influindo positivamente nos resultados obtidos. E McGraw (2005) ressalta que os testes em questão não devem ser restritos à infraestrutura, aos ativos de rede, ao sistema operacional: deve-se explorar a fundo

a aplicação, suas funcionalidades e seus componentes, o que pode demandar o uso (ou a construção) de software específico.

6. Após a conclusão do desenvolvimento, estando o software em condições de iniciar a operação, a **configuração** e a **adequação do ambiente de operação** impacta diretamente na segurança. Adequar os parâmetros do ambiente requer um profundo conhecimento detalhado dos componentes e dos requisitos do software para garantir o máximo de segurança em todos os aspectos. De modo especial os componentes de rede e o sistema operacional, assim como os parâmetros de *log* de eventos e o monitoramento de eventos são fundamentais para a segurança, além dos parâmetros do próprio software.

Um aspecto de grande importância desta proposta de McGraw (2005) é a aproximação entre as equipes e profissionais de segurança da informação e os desenvolvedores de software. O autor enfatiza que é esta integração que viabiliza a utilização da abordagem proposta e, mesmo para instituições e equipes com grande maturidade, não se trata de algo simples. Além disso pode ser necessário adaptar ou fazer ajustes ao modelo para acomodar necessidades específicas ou compatibilizar a proposta com práticas já adotadas e inovações na área de segurança da informação.

3.2 Casos de uso impróprio

Propostos originalmente por McDermott e Fox (1999) como uma forma de representar os requisitos de segurança do software de um modo simplificado, os casos de abuso – *abuse case* – ou melhor, caso de uso impróprio[3] são uma adaptação dos casos de uso da UML. McDermott e Fox (1999) apresentam os casos de uso impróprio como uma representação da interação de atores com um sistema da qual resulte um dano ao sistema, a outros atores ou aos interessados no sistema. Neste caso, os atores que provocam o dano – seja intencional ou não – são chamados de mal atores, atores maliciosos ou atacantes. Em seu estudo McDermott e Fox (1999) partiram da necessidade de representar graficamente – como mostrado na Fig. 9, de uma maneira simples e de fácil entendimento, os requisitos de segurança da informação de um software.

[3] No presente trabalho optou-se pela denominação “caso de uso impróprio” ao invés de “caso de abuso”, pois esta última denominação pode causar desconforto ao confundir-se com casos de abuso sexual ou pedofilia.

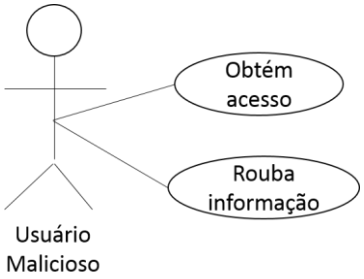


Figura 9 - Diagrama de Caso de Uso impróprio.
Adaptado de McDermott e Fox (1999).

Deste modo, usuários do software e mesmo profissionais do desenvolvimento não familiarizados com aspectos e características da segurança poderiam compreender tais requisitos. Para isso propuseram uma sequência de construção baseada na elaboração de casos de uso, sugerindo que os casos de uso impróprio sejam construídos após a elaboração dos casos de uso. Segundo esta proposição, são necessárias as seguintes etapas para a elaboração dos casos de uso impróprio:

1) A identificação dos **maus atores** (atores maliciosos ou atacantes), feita após a identificação dos atores dos casos de uso, de forma a identificar aqueles que, intencionalmente ou não, possam realizar operações que comprometam a segurança do software. Estes maus atores depois serão caracterizados em função dos recursos dos quais dispõem, de suas capacidades e de seus objetivos quanto ao uso do software.
2) A identificação das **interações** dos maus atores com o software, por meio das quais o dano pode ser concretizado. Neste ponto já é possível desenhar os diagramas de casos de uso impróprio de forma semelhante aos diagramas de caso de uso.
3) A **definição** do caso de uso impróprio, ou seja, a descrição da interação do mau ator com o software com a caracterização do dano e a explicitação dos detalhes da interação. Esta descrição poderá vir a ser revista à medida em que os casos de uso também o sejam, em função da evolução dos requisitos do software.
4) A verificação e validação da **granularidade**, cujo intuito é avaliar se há casos de uso impróprio em quantidade suficiente, isto é, que abranjam os problemas de segurança da forma mais completa possível. Também é necessário validar se as informações nos casos de uso impróprio tem o detalhamento suficiente, se não há

excessivos casos de uso impróprio ou se os casos de uso impróprio modelados incluem excessivos detalhes.

5) A verificação da **completude** e **minimalidade** dos casos de uso impróprio, ou seja, validar se cada caso de uso impróprio retrata uma possibilidade de dano ou se alguma possibilidade de dano foi esquecida ou omitida.

Para McDermott e Fox (1999) os casos de uso impróprio teriam utilidade nas principais etapas do processo de desenvolvimento de software. Na etapa de levantamento de requisitos possibilitariam aos usuários a clara identificação dos requisitos de segurança. Na etapa de projeto e teste do software os casos de uso impróprio permitiriam uma comparação e uma crítica da capacidade dos maus atores contra o nível de segurança exigido pelo software. Ainda na etapa de teste os casos de uso impróprio permitiriam a elaboração de casos de teste específicos para os requisitos de segurança. Além disso, McDermott e Fox (1999) consideram que os casos de uso impróprio poderiam auxiliar na escolha do ponto de equilíbrio entre o nível de segurança e os requisitos funcionais ou funcionalidades que representam os riscos detalhados pelos casos de uso impróprio.

A abordagem de Sindre e Opdahl (2000) acerca dos requisitos de segurança no processo de elicitação de requisitos aponta para os casos de uso impróprio – *misuse case* – como forma de apresentar os comportamentos indesejados do software no que se refere à segurança da informação. Neste trabalho a representação do caso de uso e do caso de uso impróprio é feita no mesmo diagrama.

O caso de uso seria então o modo "positivo" do requisito, isto é, o uso desejado do software, enquanto o caso de uso impróprio seria o modo "negativo" do requisito. Para destacar essa diferença os autores representam os elementos do caso de uso impróprio em negativo, como mostrado no exemplo da Fig. 10. O estudo propõe ainda que a representação possa ser de um caso de uso que mitiga o risco de um caso de uso impróprio, ou de um caso de uso impróprio que ameace um caso de uso.

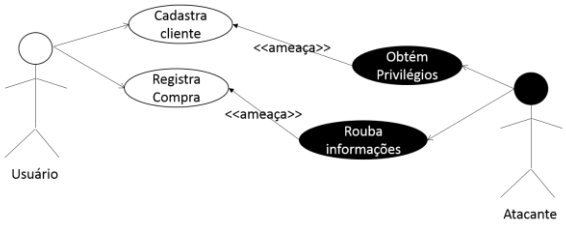


Figura 10 - Caso de uso impróprio
Adaptado de Sindre e Opdhal (2000).

De acordo com Sindre e Opdahl (2000), um caso de uso impróprio inclui também um descritivo textual composto de informações como Nome, Resumo, Autor e Data, cujo conteúdo seria o mesmo de um caso de uso, o Caminho básico e os Caminhos alternativos, que representam as operações que resultariam no dano. Além disso, outras informações como Exceções, Pontos de extensão, Gatilhos, Premissas, Pré-condições, Pós-condições, Regras de negócio relacionadas e Ameaças fazem parte do documento.

Como orientação para a elaboração dos casos de uso impróprio, o trabalho de Sindre e Opdahl (2000) apresenta um processo composto das seguintes etapas:

1) A **identificação dos ativos críticos** da organização, ou seja, informações de valor para a organização, seus locais de armazenagem e os processos que tratam tais informações;
2) A definição dos **objetivos de segurança** para cada ativo, preferencialmente de acordo com os critérios comuns (*common criteria*) para a segurança da tecnologia da informação;
3) A **identificação das ameaças** a cada objetivo de segurança, os possíveis atores e as sequências de ações que podem resultar em danos;
4) **Identificação** e **análise** d**os riscos** representados pelas ameaças, utilizando a análise de riscos e custos, típica da área de engenharia da segurança;
5) A definição dos **requisitos de segurança** ou **contramedidas** aos riscos, ou seja, a forma pela qual o software se defenderá das ameaças evitando, eliminando ou reduzindo os riscos.

Segundo Sindre e Opdahl (2000) este processo é cíclico, e deve evoluir com o decorrer do projeto para refletir as mudanças e inovações do software. Uma vantagem apresentada pela proposta é possibilitar a abordagem precoce da segurança da informação, envolvendo os interessados e usuários, além dos profissionais de segurança. Já a desvantagem é que, por ser um processo aberto, há que se cuidar para que não se estenda infinitamente, uma vez que as próprias contramedidas ou precauções podem inserir novos riscos e ameaças ao software.

Os casos de uso impróprio são efetivamente a ferramenta mais apropriada para a modelagem de requisitos “negativos”, isto é, aquilo que não se espera que aconteça com o software. De acordo com (WHITTLE et al, 2008), entretanto, a despeito de serem fáceis de construir e entender, não é simples a tarefa de testar ou analisar tais artefatos. Com a intenção

de avançar neste sentido, (Whittle et al, 2008) propuseram uma técnica baseada na UML para uma linguagem que possibilita a modelagem de casos de uso impróprio e também a sua execução, confrontando-os com os casos de uso do software.

Esta proposta facilita sobremaneira a avaliação dos casos de uso impróprio por meio de sua apresentação aos interessados em geral, permitindo a avaliação da efetividade e da abrangência destes casos e das contramedidas adotadas para enfrentá-los. O trabalho de (Whittle et al, 2008) faz uso de cenários modelados em EIODs – *Extended Interaction Overview Diagrams* – diagramas de visão geral de interação estendida, que vêm a ser uma formalização dos diagramas de interação estendida da UML, que possibilitam a modelagem precisa de cenários. Após a modelagem, os diagramas são transformados em máquinas de estado finito (FSM – *Finite State Machines*) que podem ser animadas, isto é, pode-se executar os casos de uso impróprio e avaliar a sua efetividade.

Como resultado de seu trabalho, (Whittle et al, 2008) implementaram esta proposta em uma ferramenta denominada MUCSIM – *Misuse Case Simulator* – que funciona como um *plug-in* da suíte de modelagem IBM *Rational Software Modeler*. E no seu estudo aplicaram a ferramenta na modelagem de softwares cujo aspecto de segurança era crítico, e obtiveram resultados satisfatórios. Na avaliação dos pesquisadores a solução proposta mostra-se efetiva na modelagem de casos de uso impróprio em nível de aplicação e de sistema, pois trata-se de uma abordagem ainda genérica. Esta modelagem permite uma maior precisão na elaboração dos casos de uso impróprio e também das medidas de segurança adotadas a partir destes modelos, que podem ser testadas quanto à sua efetividade.

Para a utilização bem-sucedida de casos de uso impróprio é necessária a definição ou escolha de um modelo descritivo que preserve a simplicidade e facilidade de construção e adaptação. Além disso é necessário que se consiga expressar de maneira natural as ameaças e riscos perante os requisitos de segurança do software. Com o intuito de suprir essa lacuna, (Hartong et al, 2009) apresentam um meta-modelo para a elaboração dos casos de uso impróprio.

Nesta proposta (Hartong et al, 2009) abordam um modelo gráfico para o diagrama de caso de uso impróprio que se assemelha bastante com o modelo proposto por Sindre e Opdahl (2000). Entretanto buscam diferenciar as relações de associação entre os casos de uso e os casos de uso impróprio, estendendo-as para seis tipos distintos, a saber:

1) Uma relação na qual o caso de uso **previne** a ocorrência dos problemas definidos por um caso de uso impróprio;
2) A relação na qual o caso de uso **detecta** a ativação de um caso de uso impróprio;
3) A **ameaça**, que implica no risco de um caso de uso impróprio impedir que um caso de uso seja executado completamente ou com sucesso;
4) A **mitigação**, relação na qual um caso de uso reduz a possibilidade de que um caso de uso impróprio afete a função do software à qual representa;
5) A relação na qual um caso de uso **agrava** o dano causado por um caso de uso impróprio. Esta relação também pode ocorrer entre dois casos de uso impróprio;
6) E por fim a relação de **conflita com**, na qual dois casos de uso conflitantes podem resultar em dano, configurando o comportamento de um caso de uso impróprio.

O estudo de (Hartong et al, 2009) também propõe uma descrição textual bastante próxima do modelo de Sindre e Opdahl (2000), diferenciando-se basicamente pelo campo destinado ao pior caso da ameaça – *worst case threat* – ou seja, no máximo dano causado quando o caso de uso impróprio for bem-sucedido. Como tanto o diagrama de caso de uso impróprio quanto a descrição textual são maneiras informais de descrever o comportamento do software, este estudo também propõe o uso da notação OCL – *Object Constraint Language* – linguagem de restrição de objetos, um padrão UML, para a definição das interfaces e das funcionalidades do sistema. O *template* do esquema operacional padrão do modelo proposto contém as seguintes informações:

- Operação: é a identificação do caso de uso impróprio, que manipula ou qualifica as propriedades da informação;
- Descrição: é o detalhamento da operação;
- Observações: complementa as informações acerca do caso de uso impróprio;
- Casos de uso: relaciona os casos de uso implicados ou impactados com a operação do presente caso de uso impróprio;
- Escopo: contém as limitações e a abrangência do caso de uso impróprio;
- Declarações: compreende os comandos ou ações do caso de uso impróprio;
- Envios: apresenta as saídas ou resultados da execução do caso de uso impróprio (com sucesso ou não);
- Pré-condições: O estado das informações no contexto do caso de uso impróprio antes da execução da operação;

– Pós-condições O estado das informações no contexto do caso de uso impróprio após a execução da operação.

O meta-modelo proposto tem como objetivo, além de acrescentar a formalidade da OCL aos casos de uso impróprio, reduzir a ambiguidade e as inconsistências, aperfeiçoando a especificação dos requisitos de segurança.

Em seus estudos, (Hope et al, 2004) consideram que os casos de uso impróprio auxiliam os desenvolvedores a pensar como os atacantes. Elaborando-se os casos de uso impróprio ainda na especificação dos requisitos, antecipa-se também a avaliação das ameaças e das possíveis falhas do software, e consequentemente as contramedidas de proteção. (Hope et al, 2004) destacam que a segurança não é um conjunto de funcionalidades, e sim uma propriedade geral do sistema. Em função disso a elaboração dos casos de uso impróprio deve considerar a forma de agir dos adversários do sistema, ou seja, deve-se considerar aquilo que não se deseja ou não se espera que seja feito, e como o software deverá reagir nesses casos. Uma avaliação de padrões de ataques, das fontes recorrentes de ataques e seus alvos, bem como do comportamento dos adversários e dos próprios usuários do software é uma boa premissa para a elaboração de casos de uso impróprio mais efetivos.

Avaliando a prática corrente de postergar as questões de segurança ao máximo, atentando para estas somente nas etapas finais do processo de desenvolvimento do software, Petersen e Steven (2006) destacam a necessidade de incluir os casos de uso impróprio no início do SDLC. Isto pode gerar dúvidas, uma vez que aspectos elementares do software como tipos de dados, métodos, interfaces e tecnologias ainda não foram definidos. Porém deve-se manter o foco nos aspectos conceituais dos ataques e falhas, contra os quais o software deve ser protegido. Além disso, os casos de uso impróprio não são um fim em si mesmo, devendo ser vistos como parte de um processo incremental cujo objetivo é garantir a segurança da informação.

Segundo Petersen e Steven (2006) uma boa prática na elaboração de casos de uso impróprio é partir dos caminhos alternativos e condições de erro dos casos de uso. Os casos de uso impróprio ajudam a apontar diversas ocorrências no processo de desenvolvimento, identificando as ameaças, estreitando o relacionamento entre arquitetos, desenvolvedores e profissionais de segurança da informação, caracterizando “o que” e “onde” a respeito de falhas e ataques, deixando claro ao time de projeto do software quais são os temores dos interessados e usuários quanto à segurança, orientando todos os fluxos dos processos do software à

segurança da informação e provendo um *feedback* oportuno a respeito da arquitetura de segurança do software, seus projetos e planos de teste.

Outro aspecto interessante do trabalho de Petersen e Steven (2006) é que, embora tenham avaliado ferramentas para a elaboração de casos de uso impróprio, os autores concluem que não é necessário dispor de algo específico. É possível elaborar casos de uso impróprio com uma ferramenta de diagramação simples ou genérica, um editor de textos e criatividade.

3.3 Análise de Riscos

A análise de riscos é um processo da área de conhecimento de gestão de riscos que busca identificar os riscos e apresentar estimativas, isto é, uma quantificação probabilística da ocorrência de incidentes – a ocorrência ou efetivação do risco - e o impacto destes incidentes. Esta análise - em especial a identificação de riscos – é essencial para prover suporte à elicitação dos requisitos de segurança do software, mas não deve se restringir às fases de planejamento do projeto e à análise de requisitos, uma vez que, como mostrado na Fig. 11, os riscos existem em todos os níveis e seus efeitos se propagam entre eles (Raspotnig e Opdahl. 2013).

A análise de riscos deve ser constante e recorrente no SLDC, o que não é uma tarefa das mais simples, a começar pela escolha dos métodos e técnicas a serem empregados para tal atividade. Este tópico apresenta estudos sobre a análise de riscos cujo propósito é auxiliar tanto nessa escolha quanto na efetiva execução da atividade em si.

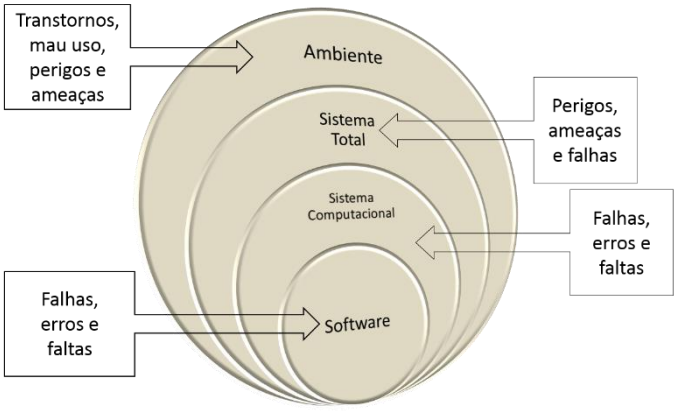


Figura 11 – Os níveis dos riscos computacionais.
Adaptado de (Raspotnig e Opdahl, 2013).

De acordo com Verdon e McGraw (2004) um dos aspectos mais importantes da segurança da informação no SDLC é a abordagem precoce. Tratar a segurança o mais cedo possível reforça a atuação preventiva quanto aos riscos aos quais o software e seu processo de desenvolvimento estão expostos, e que resultarão em problemas para a segurança da informação. Para que seja possível mitigar esses riscos e tratar as ameaças com contramedidas efetivas é necessário identificar, classificar e avaliar de forma quantitativa e qualitativa os mesmos, ou seja, efetuar a análise de riscos. Verdon e McGraw (2004) ponderam que a análise de riscos se baseia no conhecimento de vulnerabilidades, ameaças, impacto e probabilidade. As abordagens básicas são as que consideram as perdas financeiras, as classificações de riscos derivadas matematicamente e as técnicas de avaliação qualitativas. Todas estas abordagens consideram os seguintes elementos para efeito de análise:

- Os ativos ou objetos de proteção, como informações, dados ou todo o sistema;
- Os riscos, isto é, a possibilidade de ocorrência de eventos que afetem os ativos;
- As ameaças ou possíveis fontes de eventuais danos aos ativos;
- As vulnerabilidades – possíveis defeitos ou fraquezas dos ativos;
- As contramedidas ou proteções aos ativos;
- Os impactos, ou seja, as possíveis perdas em função de danos aos ativos;
- A probabilidade de ocorrência dos riscos.

Uma vez identificados e mensurados os riscos, os requisitos de segurança da informação deverão refletir as providências para contê-los. Verdon e McGraw (2004) avaliaram três ferramentas para esta representação: a SecureUML, a UMLsec e os casos de uso impróprio, e concluíram que a análise de riscos é de suma importância na fase de elaboração do projeto do software, e que o processo de análise de risco deve ser contínuo, identificando as vulnerabilidades e calculando o impacto de modo a embasar as contramedidas e a estratégia de mitigação destes riscos.

Raspotnig e Opdahl (2013) analisaram diversas técnicas de identificação de riscos – considerando a segurança dos usuários (*safety*) e a segurança de sistemas e da informação (*security*). A conclusão foi que os métodos e as técnicas voltadas para a segurança do usuário estão em um nível de maturidade bastante avançado, seja em função do emprego há longo tempo – algumas técnicas remontam às décadas de 60 ou anteriores – ou em função da própria maturidade da indústria que as implementa, usa e aprimora. Exemplos claros são as técnicas de identificação de riscos empregadas pela indústria aeronáutica e de transporte aéreo, e pelos

setores militares ligados às defesas dos países, tais como a *Functional Hazard Assessment – FHA* ou Avaliação Funcional de Perigos, a *Preliminary Hazard Analysis – PHA* ou Análise Preliminar de Perigos.

A *Functional Hazard Assessment – FHA* ou Avaliação Funcional de Perigos originou-se na indústria aeronáutica e, em função da simplicidade e da eficiência, tornou-se conhecida e usada nas demais indústrias (ARP, 1994). É uma técnica que considera as funções do sistema relacionadas a conjuntos de faltas ou perigos, dependente da profundidade do levantamento. Baseia-se em registros em planilha contendo basicamente as seguintes informações: componente, modo da falta, taxa da falta, modo de operação, fator causal, efeitos, ação ou mitigação recomendada, comentários.

Outra técnica que foi considerada, a *Preliminary Hazard Analysis* – PHA, foi desenvolvida pela área de defesa dos Estados Unidos nos anos 60 (Ericsson, 2005). Empregada para a elicitação de requisitos de segurança do sistema no projeto inicial ou nas suas especificações, tem como base uma planilha que coleta as seguintes informações: perigos do sistema, efeitos dos perigos, causas dos perigos, mitigações, levantamento inicial do risco.

Usada para identificação e analise de perigos e riscos operacionais, a *Hazard and Operability – HAZOP* é empregada pela indústria química desde a década de 70, mas aplicada em diversas indústrias (Raspotnig e Opdahl, 2013). Trata-se de técnica para identificar e analisar perigos e ameaças operacionais em um sistema. Com base em uma planilha são registrados: item, parâmetro, palavra-guia, consequência, causa, perigo, recomendações.

A técnica do *Failure Mode and Effect Analysis – FMEA* consiste em analisar falhas em potencial de um sistema e avaliar os possíveis efeitos (Raspotnig e Opdahl, 2013). Padronizada pelos militares americanos na década de 40, busca estabelecer a confiabilidade dos sistemas com base nos seguintes registros: componente, modo da falha, taxa da falha, fatores causais, efeito imediato, efeito no sistema, método de detecção, controles atuais ou ações recomendadas. É uma técnica popular devido ao fato de ser de fácil aprendizado e prática, e aplicável a diferentes tipos de equipamentos, sistemas e processos.

Por fim a *Fault Tree Analysis - FTA* é uma técnica mais sofisticada que faz uso de um gráfico em formato de árvore. Baseada na lógica booleana, busca estabelecer a causa raiz do evento analisado (Raspotnig e Opdahl, 2013). A árvore é composta de nodos (raiz, intermediário e folha), portas (E e OU) e quinas (links entre os nodos). Desenvolvida e usada

nos projetos de mísseis nucleares nos anos 60, foi incorporada pela aviação comercial e atualmente é bastante usada nas indústrias relacionadas à segurança (*safety*).

Já no que tange à segurança da informação, Raspotnig e Opdahl (2013) analisaram seis técnicas diferentes, comparando-as com base em critérios específicos. Obviamente isto não exaure a pesquisa sobre tais técnicas e a quantidade delas certamente é muito maior. Entretanto, foram consideradas as de maior relevância, principalmente no que se refere ao uso conhecido e aos trabalhos disponíveis em estudos de nível apropriado, cujas características principais são apresentadas a seguir.

O KAOS - *Knowledge Acquisition in Automated Specification* – é um *framework* com uma metodologia voltada inicialmente para engenharia de requisitos, baseada em suporte à modelagem e elaboração de requisitos com foco em metas. Os artefatos de modelagem incluem as metas, os objetos, os agentes, as operações, os requisitos e os obstáculos. Esses artefatos são utilizados para produzir um antimodelo, que tem como propósito mostrar como, por que e por quem o modelo é ameaçado, com base em contra-metas, atacantes, contra-requisitos, vulnerabilidades, atacado e contramedidas (Raspotnig e Opdahl, 2013).

Já a *Secure Tropos*, diferentemente da *KAOS*, é uma técnica voltada para todas as fases do desenvolvimento, e não apenas para a engenharia de requisitos. Porém enfatiza a abordagem precoce da engenharia de requisitos voltada para a segurança. O ponto principal é um modelo representado graficamente, que é refinado e estendido à medida em que se avança nas fases do desenvolvimento (Raspotnig e Opdahl, 2013). Este modelo consiste em atores, metas, tarefas ou planos, recurso, dependência, regra e posição. Além disso, o modelo de segurança contempla as restrições de segurança (confidencialidade, integridade e disponibilidade, por exemplo), as dependências de segurança e as entidades de segurança (metas, tarefas, recursos ou capacidades).

A técnica de casos de uso impróprio (*Misuse Case*) é uma referência aos casos de uso para criar e relatar os casos de uso impróprio - abusos, que expõem o sistema a riscos ou ameaças e o levam a ocorrência de faltas (Raspotnig e Opdahl, 2013). Os elementos dos casos de uso impróprio são os atores do abuso, as suas ações, as ameaças – isto é, como o caso de uso impróprio relaciona-se aos casos de uso, explorando suas vulnerabilidades, os casos de uso de segurança (cujo objetivo é mitigar esse abuso) e a mitigação (as contramedidas ao abuso). Os casos de uso impróprio foram abordados na Seção 3.2 deste capítulo.

O *abuse frame* ou quadro de uso impróprio é uma técnica derivada dos quadros de problemas de Jackson, que buscam estabelecer os limites do sistema, e que tem por objetivo prover um modelo abstrato das ameaças impostas por usuários maliciosos a estes limites. O quadro de problemas considera a máquina, o requisito e o mundo do problema, enquanto o quadro de uso impróprio acrescenta os contra requisitos, atacante, ativo e máquina maliciosa (Raspotnig e Opdahl, 2013).

A técnica da árvore de ataques (*Attack Tree*) é uma maneira formal e metódica de descrever a segurança dos sistemas baseadas em ataques (Raspotnig e Opdahl, 2013). Assim como a FTA, usa uma estrutura gráfica para representar os ataques ao sistema, construída com base na lógica booleana e usando dos seguintes conceitos: Nodo do topo (meta do ataque), nodos abaixo (meios para viabilizar o ataque) e portas (relações entre os nodos usando E e OU).

A técnica da árvore de ameaças (*Threat Tree*) é uma técnica analítica cujo objetivo é apoiar a análise de requisitos de segurança e seus desdobramentos em função das ameaças (Raspotnig e Opdahl, 2013). Derivada da técnica FTA, usa uma estrutura gráfica para descrever as relações entre os nodos da árvore, definidos como: nodo raiz, nodos intermediários, nodos folha e relações (E e OU).

A pesquisa de Raspotnig e Opdahl (2013) enfatiza a evolução da efetividade de tais técnicas e ressalta que, apesar da complexidade, estas técnicas vêm apresentando resultados satisfatórios em suas respectivas áreas e indústrias, e apesar disso vêm sofrendo contínuas evoluções. O trabalho utiliza critérios como tempo de uso, o envolvimento dos interessados, o tipo de sistema, a área de aplicação, a disposição em camadas ou níveis, as entradas, o processo e as saídas produzidas, a interoperabilidade, a escalabilidade, a clareza e a forma de comunicação de cada método e técnica avaliada, e faz inúmeras comparações entre elas com base naqueles critérios. Isto torna este trabalho de grande valia para a escolha de um método ou técnica de avaliação de ameaças ou riscos nas etapas iniciais do processo de desenvolvimento de software.

3.4 Engenharia de Requisitos de Segurança

Uma vez identificados os riscos e as ameaças, cabe ao processo de desenvolvimento do software empregar um método, técnica ou procedimento para transformar as medidas e contramedidas de segurança em requisitos de segurança. Existem vários métodos e técnicas, muitas das quais

já consolidadas, porém de uso restrito, e muitas ainda carentes de maturidade ou comprovação de efetividade, de acordo com Salini e Kanmani (2012), em estudo no qual apresentam uma avaliação destes métodos e técnicas, entre as quais:

- SQUARE – *Security Quality Requirement Engineering*, do SEI – *Software Engineering Institute*;
- Haley *framework;*
- SRE - *Security Requirement Engineering* de Gustav Boström;
- CLASP – *Comprehensive, Lightweight Application Security Process;*
- Microsoft SDL - *Trustworthy Computing Security Development Life Cycle;*
- O método de Axelle Apvrille e Markan Pourzandi;
- O SREP – *Security Requirement Engineering Process*;
- O Secure Tropos.

Salini e Kanmani (2012) comparam esses métodos com base em diversos critérios, como as atividades da fase de requisitos, mostradas no Quadro 3, e as atividades do método, mostradas no Quadro 4. Em sua conclusão Salini e Kanmani (2012) consideram que, mesmo que as ferramentas baseadas no SQUARE e o método SREP sejam as mais abrangentes em termos de engenharia de requisitos de segurança, o importante é adotar o uso de uma delas o quanto antes no processo de desenvolvimento do SDLC, e também prosseguir com a abordagem por todo o ciclo de vida do software.

Quadro 3 - Métodos SRE x Atividades da fase de requisitos.
Adaptado de Salini e Kanmani (2012).

Métodos	Definições	Objetivos	Uso indevido / Modelagem de ameaças	Ativos	Padrões de Codificação	Categorizaçã e Priorização	Validação	Planejamento
SQUARE	X	X	X			X	X	
SREP	X	X	X	X		X	X	
Haley		X	X	X			X	
Bostöm			X	X	X	X		
CLASP		X					X	
Secure Tropos		X			X			
Microsoft		X		X				
Apvrille e Pourzandi		X	X	X			X	X
Fernandez			X	X				
Van Wyk e McGraw			X	X				
Peterson			X	X				

Quadro 4 - Comparação entre atividades dos métodos SER.
Adaptado de Salini e Kanmani (2012).

Atividades para S R E	Atividades importantes incluídas				Uso da Engenharia de Requisitos de Segurança (S R E)		Ferramenta disponível?
Métodos S R E	Modelagem de ameaças	Análise de Riscos	Especificação	Revisão / Inspeção	Fase posterior do SDLC	Uso nas Organizações	
McGraw	Não	Sim	Sim	Não	Testes	Usado	Não
SQUARE	Sim	Sim	Sim	Sim	Projeto, Testes	Usado	Sim
Microsoft SDL	Não	Sim	Sim	Não	Não Usado	Apenas Microsoft	Não
Secure Tropos	Não	Não	Sim	Não	Não Usado	Não informado	Sim
Charles Haley	Sim	Não	Sim	Sim	Não Usado	Não informado	Não
Apvrille e Pourzandi	Sim	Sim	Sim	Não	Não Usado	Não informado	Não
Gustav Boström	Não	Não	Sim	Sim	Não Usado	Não informado	Não
CLASP	Sim	Sim	Sim	Sim	Testes	Usado	Não
SREP	Sim	Sim	Sim	Sim	Testes	Não informado	Não
Eduardo Fernandez	Sim	Não	Sim	Não	Testes	Não informado	Não
Gunnar Peterson	Sim	Não	Sim	Não	Testes	Não informado	Não

3.5 Testes de segurança da informação

A área de testes de software é objeto de recorrentes estudos, notadamente na abordagem proposta pelo presente trabalho, voltado para a segurança da informação. Como a prática corrente carrega sobre essa atividade – o teste de software – os maiores esforços para a verificação e validação da segurança da informação, há uma busca para melhorar o processo de teste, quer seja na redução dos esforços, tempo e custos dispendidos, quer seja no aprimoramento da cobertura e na efetividade, passando-se pela automatização dos testes e evolução do ferramental empregado, inclusive os modelos estatísticos (McGraw, 2005).

Entre os trabalhos e estudos voltados ao teste de software com foco na segurança da informação pesquisados encontram-se diversos que apresentam uma convergência com a proposta do presente trabalho, os quais são apresentadas na sequência. A aplicação de técnicas de teste baseadas em software de busca é avaliada por Antoniol (2009), abordando as dificuldades para – ou mesmo impossibilidade de - garantir a completa segurança da informação no software, normalmente devido ao tamanho, complexidade, extensibilidade e conectividade.

Devido à pressão pela redução de custos de produção e pela redução do prazo para a entrega do software, a fase de testes acaba sendo comprometida. A alternativa proposta por Antoniol (2009) busca oferecer um menor custo por meio de um conjunto de testes adaptável ao software, incluindo a busca por vulnerabilidades conhecidas e com maior risco para o tipo

de software que está sendo testado, no que há uma concordância com os propósitos do presente trabalho.

Também considerando aspectos de custo e complexidade dos testes voltados para a segurança da informação no software desenvolvido, Chandramouli e Blackburn (2004) propõem uma ferramenta de teste de segurança funcional com base na modelagem formal e nas informações de interface, oferecendo um framework de automação de testes (TAF – *Test Automation Framework*) e uma ferramenta baseada nesse framework (TAF-SFT - *Test Automation Framework – Security Funcional Testing*) usando a linguagem formal SCR – *Software Cost Reduction*.

Propondo uma abordagem mista para a solução do problema do oráculo em testes de segurança da informação para softwares de missão crítica, (Dong et al, 2013) fazem uma combinação de testes baseados em análise de caminho com testes de mutação, reduzindo assim o esforço de teste e mantendo a cobertura dos testes em um nível satisfatório.

Uma discussão interessante é a avaliação das próprias ferramentas de teste de segurança da informação e dos métodos propostos pelas mesmas, trabalho desenvolvido por (He et al, 2008). Os autores do trabalho avaliam o rápido desenvolvimento da tecnologia da computação confiável e do software que a suporta (TCSS - *Trusting Computer Supporting Software*), e propõem um modelo de teste e um protótipo de sistema para validar a aderência às especificações do TCG – *Trusting Computer Group*. O modelo proposto permite então o gerenciamento dos produtos bem como auxilia na escolha entre produtos classificados como aderentes à especificação de computação confiável.

No que concerne às dificuldades encontradas nas áreas de segurança da informação, qualidade de software e testes de software, uma que é a causa recorrente de problemas é a taxonomia. Observando essa questão (Hui et al, 2010) propõem uma taxonomia com base em relatos de defeitos de segurança obtidos em entidades confiáveis que registram tais defeitos, com o intuito de auxiliar os profissionais de teste de segurança na busca por tais defeitos e também a manter e expandir a lista conforme o padrão adotado, formando uma base de conhecimento uniforme.

A automatização do processo de teste com base em métodos conhecidos e o emprego sistemático de ferramentas de teste voltadas para a segurança da informação são abordadas por (Karppinen et al, 2007), que propõem uma abordagem de teste de segurança da informação com base nos objetivos de segurança, nos requisitos de segurança e nos padrões de segurança, e a

sua aplicação em um projeto real, expondo assim a enorme distância entre o modelo teórico de especificação de requisitos de segurança e as vulnerabilidades de segurança encontradas na prática, o que vem a ser um grande desafio na área. Tratando da geração de teste com uma abordagem para teste em nível de código binário para arquiteturas específicas (Li et al, 2009) propõem uma técnica de geração de teste dinâmicos no nível de código binário baseada em uma ferramenta (ReTBLDTG – *ReTargetable Binary-Level Dynamic Test Generation*) que faz uso de meta-instruções definidas (MetaISA – *Meta Instruction Set Architecture*), as quais permitem a portabilidade para outras arquiteturas de processador. No teste realizado com a ferramenta foi obtido o resultado de 100% de identificação de defeitos conhecidos em aplicações, em teste com quatro diferentes arquiteturas de processador, o que é algo promissor.

Uma proposta de geração automatizada de testes de segurança da informação com base em modelos formais de ameaças foi apresentada por (Xu et al, 2012a). A abordagem utilizada foi o mapeamento de ameaças comuns que exploram comportamentos não intencionais e entradas inválidas como estabelecido pelo modelo STRIDE – acrônimo para *spoofing identity, tampering with data, repudiation, information disclosure, denial of service, and elevation of privilegie* (roubo de identidade, adulteração de dados, repúdio, divulgação de informações, negação de serviço e elevação de privilégio) e, por meio de um mapeamento de modelo de implementação, foram criados códigos executáveis e seus mutantes para explorar as vulnerabilidades. Essa geração, assim como o código, foi executada automaticamente, e o resultado obtido foi a morte da maioria dos mutantes gerados, o que é positivo.

O registro de recursos (ativos) nos diagramas de sequência para testes de segurança da informação em aplicações web, com o objetivo de possibilitar a geração automática de casos de teste, é a proposta de (Xu et al, 2012b), cuja abordagem preconiza a mescla do mapeamento de recursos e riscos com os diagramas de sequência, possibilitando a rastreabilidade dos riscos inerentes a cada caso de uso. Essa abordagem concentra-se nos recursos utilizados/explorados por usuários legítimos e atacantes das aplicações web para efetivar o teste de segurança.

Em seu trabalho sobre teste de segurança de software orientado a dados, (Hong et al, 2012) apresentam um método cuja abordagem é voltada para o controle de acesso a dados ao invés da abordagem tradicional voltado para o controle de acesso com papéis, permissões e contextos. Segundo apurado pelos pesquisadores, essa abordagem, embora experimental, mostrou que o método é mais instrutivo e facilita a geração automática de testes.

Uma abordagem pragmática baseada em risco voltada para a validação da segurança de sistemas empresariais é proposta em trabalho de (Murthy et al, 2009). A proposta busca prover a cobertura necessária sem pressionar os custos e o esforço de teste. Os autores da pesquisa integraram as melhores práticas de segurança da informação com a abordagem baseada em risco como forma de buscar o equilíbrio entre controles de segurança e usabilidade, viabilizar a aplicação de princípios de teste baseados em risco no teste de segurança de aplicações e reduzir o tempo de projeto e os custos do desenvolvimento.

Em seu trabalho voltado para o uso de teste com base no modelo SaaS – *Software as a Service*, (Riungu et al, 2010) apresentam uma pesquisa abrangendo onze corporações. No intuito de mensurar a aparente crescente demanda de serviços de teste on-line, mensuraram os benefícios em termos de redução de custos e flexibilidade, avaliaram a consonância com serviços na nuvem e identificaram os problemas relacionados ao tratamento dos dados dos teste e à necessidade de capacitação do pessoal de teste.

No seu artigo sobre teste de segurança, Türpe (2008) propõe um aprofundamento da pesquisa no campo de teste de segurança da informação, pontuando questões até então não respondidas ou respondidas insatisfatoriamente, de modo que possibilitassem influenciar o mundo real. O autor relata sua vasta experiência em teste de invasão, sua visão crítica quanto à automação desses testes, e propõe uma agenda para pesquisa com foco na teoria das vulnerabilidades, na teoria do teste de segurança e em ferramentas e técnicas voltadas para o teste de segurança. Em contato com o autor foi verificado que, apesar de reconsiderar suas restrições quanto à automação do teste de segurança, o mesmo sustenta suas ponderações quanto às demais questões apresentadas à época do trabalho.

3.6 Considerações do Capítulo

Os diversos trabalhos aqui apresentados cobrem várias etapas do processo de desenvolvimento de software em atividades críticas para a segurança da informação, e certamente abrangem uma pequena amostra deste vasto segmento. Desde o início da era da comunicação global, enfatizada pelo advento da Internet, a preocupação com aspectos de segurança vem tornando-se proeminente. Nas últimas décadas, com o uso dos dispositivos móveis, este tema é recorrente e resulta em uma profusão de estudos e pesquisas que torna complexa a tarefa de identificar e classificar o material.

Entretanto pode-se constatar que são poucas as abordagens que apresentam o tratamento da segurança da informação de forma holística e abrangendo todo o SDLC. A grande maioria dos estudos é voltada para as etapas de teste e de codificação, sendo a especificação de requisitos e as etapas de análise e projeto deixadas em segundo plano. A utilização de uma base de conhecimentos de ameaças e falhas, a integração entre as equipes de segurança da informação e de desenvolvimento e o tratamento precoce das questões de segurança ainda são incipientes, com relativamente poucos trabalhos apresentados. No próximo capítulo será apresentada a proposta resultante do presente trabalho, a qual tem por finalidade justamente contribuir no sentido de suprir essas deficiências.

Capítulo 4 – A abordagem de Desenvolvimento Dirigido pela Segurança

Este capítulo apresenta a abordagem de desenvolvimento de software dirigido pela segurança, o modelo SDD – *Security Driven Development*. São apresentados os principais elementos do modelo, a composição destes elementos e a sua aplicação na sequência das atividades durante o SDLC.

4.1 O modelo SDD – Desenvolvimento Dirigido pela Segurança

O modelo complementar de Desenvolvimento Dirigido pela Segurança é baseado na proposta do BSIMM – *Build Security In Maturity Model* (Verdon e McGraw, 2004), (McGraw, 2005), acrescentando e adequando atividades e técnicas para aprimorar a segurança da informação no SDLC para dispositivos móveis. O SDD leva em consideração a possibilidade de uso concomitante com qualquer modelo de desenvolvimento utilizado, o uso de ferramental simples e compatível com diversos ambientes computacionais e o mínimo impacto possível no custo, no esforço e no prazo de desenvolvimento do software.

As principais características do SDD são:

- A abordagem precoce da segurança da informação no SDLC, iniciando já no processo de levantamento de requisitos e até mesmo antes, considerando-se a necessidade de uma base de conhecimentos de ameaças e vulnerabilidades que antecede ao projeto do software;
- A atuação integrada entre a equipe de desenvolvimento de software e a equipe de segurança da informação em todas as etapas do SDLC;
- O foco na segurança da informação no âmbito do software de aplicação e em seu uso;
- A utilização dos casos de uso impróprio para fins de análise de risco, planejamento e especificação de testes;
- A especificação e a construção de máquinas de ataque para a execução dos testes de segurança da informação;

– A aplicação da análise de riscos em todas as etapas do SDLC, e mais especificamente nas etapas de projeto, construção e validação do software.

O SDD enfrenta os principais riscos decorrentes do uso e do processo de desenvolvimento do software, em consonância com o que foi apresentado na Seção 2.3, na qual apontam-se estes dois elementos como origem da maioria dos problemas de segurança da informação em dispositivos móveis. Para fazer frente à esta e outras questões, abordadas no decorrer deste trabalho, a proposta do modelo SDD prioriza a segurança da informação em todo o ciclo de vida do software, desde as primeiras atividades que objetivam a caracterização da necessidade e do produto, até a sua manutenção durante o uso, como mostrado na Fig. 12.

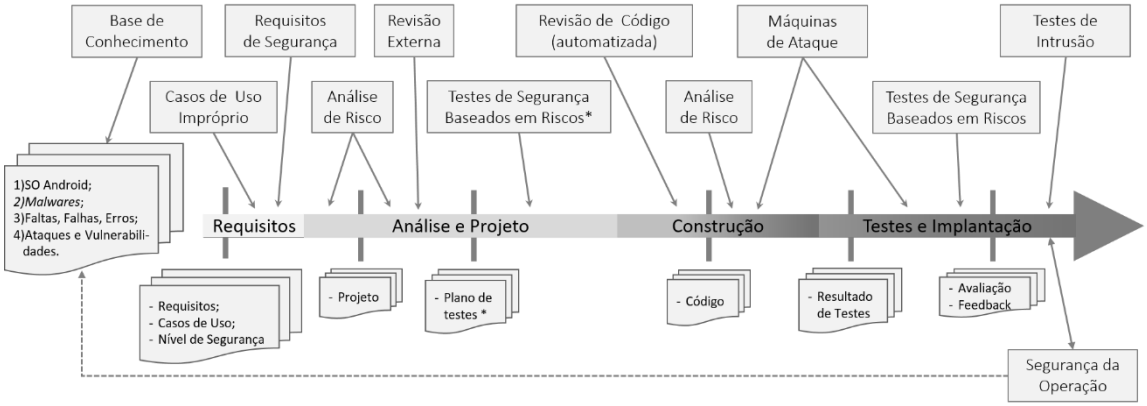


Figura 12 - O modelo SDD.

Um importante aspecto do modelo proposto é a sua aderência aos principais modelos adotados no SDLC[4], tendo como pontos de atenção a complexidade dos softwares e a avaliação do nível de segurança requerido pelo usuário. Ou seja, parte-se da premissa de que seja possível utilizar o modelo SDD no desenvolvimento de qualquer tipo de *software* para dispositivos móveis por meio de quaisquer modelos de SDLC adotado.

Nos tópicos a seguir são apresentados os elementos do modelo SDD, sua adequação às fases do SDLC, *templates* de documentos ou artefatos e exemplos de sua utilização. A Fig. 13 apresenta a visão completa do modelo SDD com destaque para as atividades e entregas enfatizadas pelo modelo durante as etapas do SDLC.

[4] Um estudo sobre os pricipais modelos é apresentado no Apêndice B.

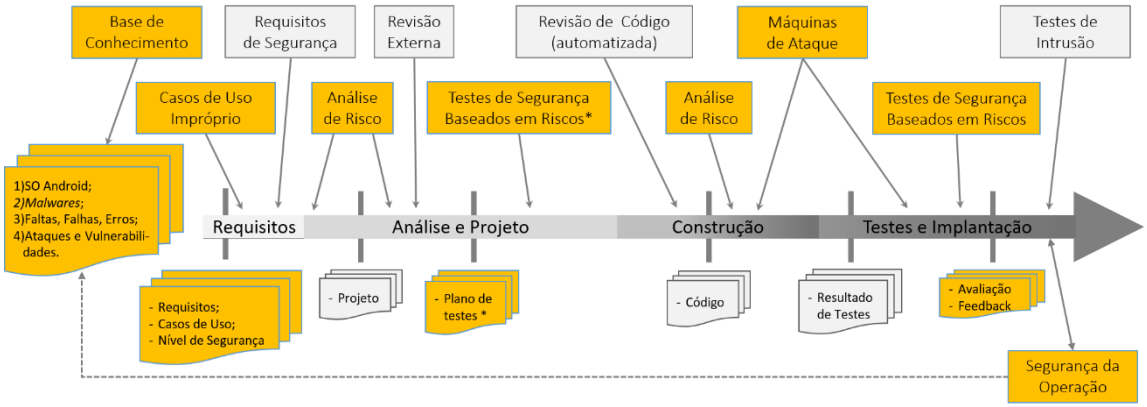


Figura 13 - Modelo SDD e os artefatos, as atividades e as entregas.

Um resumo do modelo com a relação das atividades incorporadas ou adaptadas ao SDLC, os produtos esperados em cada fase e os objetivos destas ações é apresentado na Tabela 3, e na sequência são detalhadas cada uma das atividades propostas pelo modelo SDD.

Tabela 3 - Resumo das atividades e entregas das fases no SDD.

Etapa, Fase ou Atividade do SDLC	**Ação**	**Documento, Ferramenta ou Mecanismo**	**Objetivo**
Planejamento	Classificação	Declaração do Nível de Segurança Requerido, Base de Conhecimento.	Estabelecer o nível de segurança e as implicações no SDLC e posteriores.
Especificação de Requisitos	Criação	Base de Conhecimento, Casos de Uso impróprio, Requisitos de Segurança, Análise de Riscos.	Estabelecer os limites de segurança, identificar as vulnerabilidades e os ataques conhecidos, as probabilidades de ocorrência, os custos envolvidos, os mecanismos de proteção e o retorno do investimento.
Análise e Projeto	Criação e Revisão Externa	Plano de Testes, Casos de teste de Segurança baseados nos riscos,	Definir os procedimentos de testes e as técnicas e ferramentas a serem empregadas.

Etapa, Fase ou Atividade do SDLC	Ação	Documento, Ferramenta ou Mecanismo	Objetivo
		Máquinas de Ataque.	Escrever os Casos de teste de Segurança com base nos Casos de Uso impróprio, nos Requisitos de Segurança, na Análise de Riscos e conforme o Plano de Testes. Projetar as máquinas de ataque a serem desenvolvidas e utilizadas nos testes.
Codificação / Construção	Criação, Revisão e Aplicação	Máquinas de Ataque, Casos de Uso impróprio, Casos de teste, Revisão de Código.	Efetuar os testes de segurança e avaliar a efetividade dos mecanismos e contramedidas adotadas. Atualizar os documentos e adequá-los à realidade e ao momento do projeto do software.
Testes	Revisão e Aplicação	Casos de Uso impróprio, Casos de teste, Máquinas de Ataque, Análise de Riscos.	Efetuar os testes de segurança e avaliar a efetividade dos mecanismos e contramedidas adotadas. Executar os testes de invasão e exploração de vulnerabilidades. Atualizar os documentos e adequá-los à realidade e ao momento do projeto do software. Rever a Análise de Riscos e atualizar os parâmetros. Avaliar a redução de riscos obtida.

Etapa, Fase ou Atividade do SDLC	Ação	Documento, Ferramenta ou Mecanismo	Objetivo
Operação	Revisão e Aplicação	Casos de Uso impróprio, Requisitos de Segurança, Análise de Riscos, Base de Conhecimento	Identificar as novas vulnerabilidades e os ataques sofridos. Avaliar os mecanismos e as contramedidas de proteção. Avaliar os resultados e a redução dos riscos e dados obtidos. Atualizar a base de conhecimento com as informações obtidas com o uso do software.

4.2 Nível de Segurança Requerido

A declaração de Nível de Segurança Requerido é a formalização da necessidade de proteção da informação tratada pelo software em desenvolvimento. Esta necessidade deve ser descrita com base na legislação, normas, regulamentos ou política de segurança, e sinaliza para a equipe de projeto qual é a expectativa dos usuários e dos clientes no que se refere à segurança da informação manipulada pelo software.

Também é necessário que a declaração torne explícita a responsabilidade pela segurança da informação e apresente as penalidades cabíveis em caso de descumprimento dos preceitos estabelecidos, e relacione os canais para a comunicação de incidentes de segurança da informação durante o SDLC, para o envio de avaliações e para o feedback após a instalação e implantação.

4.3 Base de conhecimento

Em seus escritos sobre a arte da guerra, o general Chinês Sun Tzu, do sec. IV A.C. anotou uma importante avaliação sobre o conhecimento:

> “Aquele que conhece o inimigo e a si mesmo, lutará cem batalhas sem perigo de derrota; para aquele que não conhece o inimigo, mas conhece a si

> mesmo, as chances para a vitória ou para a derrota serão iguais; aquele que não conhece nem o inimigo e nem a si próprio, será derrotado em todas as batalhas. ” (Tzu e Pin, 2014).

O conhecimento das características técnicas, capacidades e restrições do ambiente operacional é um diferencial importante no processo de desenvolvimento de software. Embora a indústria de software busque um modelo de desenvolvimento unificado para as diversas plataformas, sob a ótica da segurança é necessário considerar aspectos próprios de cada ambiente e arquitetura, recursos e limitações. O modelo SDD requer uma atividade complementar voltada para a manutenção de uma base de conhecimento atualizada, como um complemento ao processo de desenvolvimento. O objetivo desta base de conhecimento é disseminar, para todo o time de projeto, o conhecimento do sistema operacional, das características do ambiente de dispositivos móveis e principalmente das ameaças e vulnerabilidades às quais o produto de software estará exposto.

A formação e manutenção de uma base de conhecimento sobre as ameaças ao software é necessária para dar às equipes de desenvolvimento o entendimento do desafio a ser enfrentado. Com base neste conhecimento deve-se procurar projetar e construir software que responda adequadamente aos desafios identificados, de acordo com as expectativas dos clientes e usuários quanto à segurança da informação manipulada pelo software.

Os modelos do SDLC empregados pela indústria do software (abordados no Apêndice B) não apresentam uma atividade correspondente na fase de requisitos. O modelo SDD requer, nesta fase, o acesso a uma base de conhecimento sobre o sistema operacional e sobre as ameaças à segurança da informação em geral e ao software em particular. Esta base de conhecimento precisa agregar as informações obtidas de modo sistemático - por meio de pesquisa à literatura – mas também daquelas decorrentes da experiência da organização e das equipes de desenvolvimento.

Neste processo inicia-se a integração preconizada por McGraw (2005) e enfatizada pelo modelo SDD. Geralmente o conhecimento sobre ameaças em geral, sobre a infraestrutura e sobre a experiência dos usuários de software está no domínio das equipes e profissionais de segurança da informação. Porém a capacidade de analisar código e dados, estudar o comportamento e até mesmo realizar engenharia reversa com o intuito de obter informações relevantes é mais afeito às capacidades dos times de desenvolvimento. Portanto, o trabalho de

manutenção da base de conhecimento deve ser realizado pela atuação conjunta destes dois times.

Este conhecimento deve ser gerenciado, classificado e formatado, além de franqueado ao acesso de todos aqueles que possam contribuir para sua manutenção, e principalmente aos que dele farão uso, quer seja para sua capacitação quer seja para a busca de informações que subsidiem os projetos de software em elaboração. Com o passar do tempo esta base se tornará não apenas um importante referencial para as equipes de projetos de desenvolvimento de software e de segurança da informação, mas também uma ferramenta para o processo de capacitação das equipes, quer seja no conhecimento das ameaças e vulnerabilidades, quer seja no processo de busca, aquisição e classificação de informações sobre as mesmas.

4.3.1 Fontes de informação

No Capítulo 2 foram abordados diversos aspectos da segurança da informação para os ambientes de dispositivos móveis, com o propósito de subsidiar justamente esta demanda do modelo SDD: obtenção de informação confiável e atualizada sobre as mais diversas ameaças ao software que se pretende construir.

É importante ressaltar aqui a diferença em relação ao senso comum sobre ameaças e a abordagem proposta pela SDD: com suporte nos conceitos de dependabilidade e confiabilidade de (Avizienis et al, 2004), a base de conhecimento leva em consideração também as ameaças decorrentes do ambiente computacional – incluindo-se os serviços de rede – e do sistema operacional. Durante a elaboração da proposta do SDD foram utilizados os trabalhos de (La Polla et al, 2013) no que se refere ao ambiente *de dispositivos móveis*, o de (Braga et al, 2013), referente ao sistema operacional Android, e os de (Mylonas et al, 2011), (Shabtai et al, 2010) e (Zhou e Jiang, 2013) no que tange a Android *malwares*, os quais são bons referenciais em termos de fontes de informação para a base de conhecimento.

Outras fontes de informação sobre ameaças e vulnerabilidades consultadas: os fabricantes de dispositivos móveis e os voltados para a segurança da informação e infraestrutura, os *web sites* de comunidades, grupos e instituições ligadas à segurança da informação, como o OWASP (em www.owasp.org), uma iniciativa que mantém pesquisas sobre identificação, prevenção e estudo das vulnerabilidades em geral, o CERT-BR (www.cert.br) - Centro de Estudos, Resposta e Tratamento de incidentes de segurança no Brasil, e a F-Secure Corp. (https://www.f-secure.com/en/web/labs_global/threat-descriptions),

que contempla pesquisas, *white papers* e uma lista atualizada com as características técnicas e instruções de remoção de ameaças maliciosas identificadas por seus pesquisadores.

Alguns sites de fornecedores de produtos de segurança e infraestrutura também foram utilizados, como por exemplo:

- Google: https://source.android.com/devices/tech/security
- Avira: www.avira.com.br;
- Symantec: www.symantec.com;
- McAffee; www.mcafee.com/br/products/mobile-security
- PSafe: http://www.psafe.com/blog/
- Cisco: www.cisco.com
- Hauwei: em www.huawei.com

4.3.2 Ferramentas

Para a criação e manutenção de uma base de conhecimento sobre ameaças e vulnerabilidades é possível se valer de uma ampla gama de ferramentas, dentre as quais cabe citar:

- As Wikis, estruturas da Internet e intranet criadas a partir do início da década de 1990, que são um tipo de software colaborativo que abre a possibilidade da criação e edição de documentos em uma sistemática que dispensa a revisão antes da publicação. Muitas empresas as utilizam como ferramenta de gestão e divulgação do conhecimento;
- Os blogs, páginas que possibilitam a publicação de conteúdo em tópicos de forma incremental e dinâmica, além de permitir a interação e a pesquisa. A construção é simples e o uso é bastante comum entre os profissionais de tecnologia da informação;
- Um Portal, voltado para a gestão de conhecimento, é uma prática recorrente nas organizações. Geralmente carrega as características de uma página típica da Internet, porém com uma abordagem mais simples, controle de acesso e de postagens, e com uma temática bem definida;
- A Intranet, um poderoso veículo de comunicação e interatividade das organizações, tem características semelhantes às dos portais, porém com a possibilidade de acesso a diversas ferramentas, incluindo-se aquelas típicas de tarefas de escritório, como

editor de texto, planilhas eletrônicas, apresentações e bancos de dados. Uma das vantagens da Intranet é a integração com o ambiente, possibilitando a administração do uso e o acesso a ferramentas mais elaboradas, como as utilizadas em gestão de projetos, edição e versionamento de documentos e também de worklflow;

- Os Grupos ou Listas de discussão, mecanismos fornecidos por servidores de correio eletrônico e alguns provedores de *webmail* que fazem uso do e-mail para compartilhar mensagens e permitir a interação, com a possibilidade de arquivamento e recuperação das mensagens e da moderação ou filtragem de conteúdo.
- As mídias sociais, como Twitter, Facebook, Linkedin, entre outros, também são de grande valor para a obtenção e o compartilhamento de conhecimento e informações.
- As ferramentas de administração de dados, voltadas para BI – *Business Inteligence* e *Big Data* podem ser consideradas em função do volume de dados da base de conhecimento, sua abrangência e distribuição geográfica e o uso de diversas fontes de informação.

Mais importante do que a escolha da ferramenta é a validação da informação, sua classificação e formatação, que devem ser conduzidas de modo que favoreçam a busca, o acesso e o entendimento. Do mesmo modo a atualização e a manutenção, que garanta a informação atualizada e integra. Também é importante garantir que a informação esteja disponível da forma mais abrangente possível, especialmente para os ambientes de dispositivos móveis. Para isto é necessário um planejamento para a taxonomia, o uso de ferramental que possibilite a indexação e a busca, e que permita uma administração simplificada, porém capaz de garantir a segurança das informações.

4.3.3 Modelos

No Capítulo 2 foram apresentados alguns estudos que abordam o sistema operacional Android e que são um bom referencial para a base de conhecimento acerca deste sistema. A disponibilização deste material já é um bom começo para o processo de capacitação, formação e atualização de equipes para a atuação em projetos de desenvolvimento de software para ambiente de dispositivos móveis.

No que se refere às ameaças e vulnerabilidades, ainda no Capítulo 2 as Seções 2.2 e 2.5 apresentam estudos detalhados sobre a classificação e a documentação destas, oferecendo

informações e modelos a ser utilizados para a base de conhecimento e sua estruturação, sendo parte da contribuição do presente trabalho para a implementação do modelo SDD.

4.4 Casos de uso impróprio

Para enfatizar a abordagem precoce da segurança da informação no SDLC, o modelo SDD inclui, durante a preparação dos requisitos (levantamento, elicitação, etc.) e elaboração dos casos de uso, a criação dos casos de uso impróprio. Como abordado no Capítulo 3, o insumo para a elaboração dos casos de uso impróprio são os casos de uso e os requisitos de segurança.

O modelo SDD adota a abordagem baseada no meta modelo de (Hartong et al, 2009). E observa as recomendações de Petersen e Steven (2006) quanto ao ferramental a ser empregado, valendo-se de um editor de texto e uma ferramenta simples de diagramação ou de desenho para a elaboração dos casos de uso impróprio. O propósito é facilitar e simplificar o processo de construção dos casos de uso impróprio, enfatizando a abordagem precoce e reforçando a integração da atuação das equipes de segurança da informação e de desenvolvimento de software.

Entretanto, como o foco é promover a abordagem precoce da segurança da informação e explicitar as vulnerabilidades e ameaças, a opção pelo modelo citado não é incondicional, e destina-se a apresentar um referencial e proporcionar a orientação para a correta implementação do modelo. O modelo SDD nesta fase reforça a sequência adequada do processo, promovendo a elaboração dos casos de uso impróprio após a elaboração dos casos de uso – os quais podem ser usados como subsídio para aquela elaboração.

Convém reforçar que, enquanto um caso de uso visa explicitar uma ação relacionada a um requisito, a uma funcionalidade do software, um caso de uso impróprio tem por finalidade apresentar uma situação em que um dano é provocado em função de uso incorreto ou malicioso do software, a partir de um conjunto de funcionalidades ou dos objetivos de um usuário ou mesmo de um atacante ou usuário mal-intencionado. Desta forma um caso de uso impróprio pode abranger mais que uma funcionalidade, ou um conjunto delas, bem como retratar um mal funcionamento decorrente de várias iniciativas ou ações, porém com o mesmo resultado prático.

4.4.1 A elaboração dos casos de uso impróprio

O modelo SDD requer a elaboração dos casos de uso impróprio após a elaboração dos casos de uso do software, com os requisitos já identificados, especialmente os requisitos de segurança da informação.

Na primeira etapa deve-se identificar e caracterizar o ator ou atores do caso de uso impróprio, analisando sua capacidade e os recursos dos quais dispõem para classificar seu grau de periculosidade. A análise criteriosa do ambiente e do sistema operacional que suportará o software é muito importante, requerendo a atenção e o interesse do pessoal de segurança da informação. Nesta etapa conta-se também com a participação dos usuários do software ou dos clientes do projeto.

Na segunda etapa, uma vez feita a identificação e a caracterização dos atores, são identificadas as interações destes atores com o software de modo a causar danos. Nessa etapa são enumeradas as prováveis interfaces do software e as ameaças e vulnerabilidades tornam-se mais visíveis, possibilitando o desenho dos diagramas de casos de uso impróprio.

Uma vez definidas as interfaces do software e identificados os principais os componentes, procede-se a terceira etapa do processo, na qual se escreve o caso de uso impróprio propriamente dito. O SDD estabelece o uso de um diagrama da estrutura do software - em árvore, por exemplo – para identificar os possíveis pontos de ocorrência do uso impróprio. Pode ser necessário aguardar definições mais elaboradas do software, e isso implica em refinamentos posteriores do documento. O processo de refinamentos sucessivos é adequado tal qual na elaboração dos casos de uso.

Feitas estas descrições, na quarta etapa considera-se a abrangência dos casos de uso impróprio e o nível de detalhe das informações com base na Declaração de Nível de Risco, levando-se em conta a finalidade do software. Softwares nos quais a segurança é um aspecto crítico certamente demandarão mais esforço, enquanto que para softwares com requisitos de segurança da informação mais simplificados se evita um número excessivo de casos de uso impróprio, e também que os casos de uso impróprio sejam demasiadamente detalhados.

Uma atenção especial nessa etapa é a identificação daqueles casos de uso impróprio que são possíveis, porém improváveis. Casos de uso impróprio muito similares ou cuja diferenciação seja possível apenas por pequenos detalhes também podem ser unificados ou sumarizados, e deve-se evitar casos de uso impróprio demasiadamente complexos ou abstratos.

Um bom caso de uso impróprio deve manter uma uniformidade com os demais, podendo ser necessário um refinamento ou mais abstrações para obter essa uniformidade.

A quinta e última etapa é uma verificação final do processo de elaboração, cujo intuito é verificar a integridade do caso de uso impróprio e confirmar se o mesmo apresenta o mínimo possível de abrangência para o propósito a que se destina, quer seja, descrever uma interação com o software que venha a produzir danos para este, para o usuário do software ou outros interessados em seu resultado.

A verificação também deve buscar identificar possíveis negligências, o que pode ser feito revisitando os requisitos e os casos de uso juntamente com funcionalidades e requisitos de segurança. Validar com os usuários ajuda a evitar que falte algum aspecto ou que seja esquecido algum possível abuso. A Fig. 14 apresenta um fluxograma do processo descrito. Um exemplo da utilização deste modelo no estudo de caso é apresentado na Seção C.1.1 do Apêndice C.

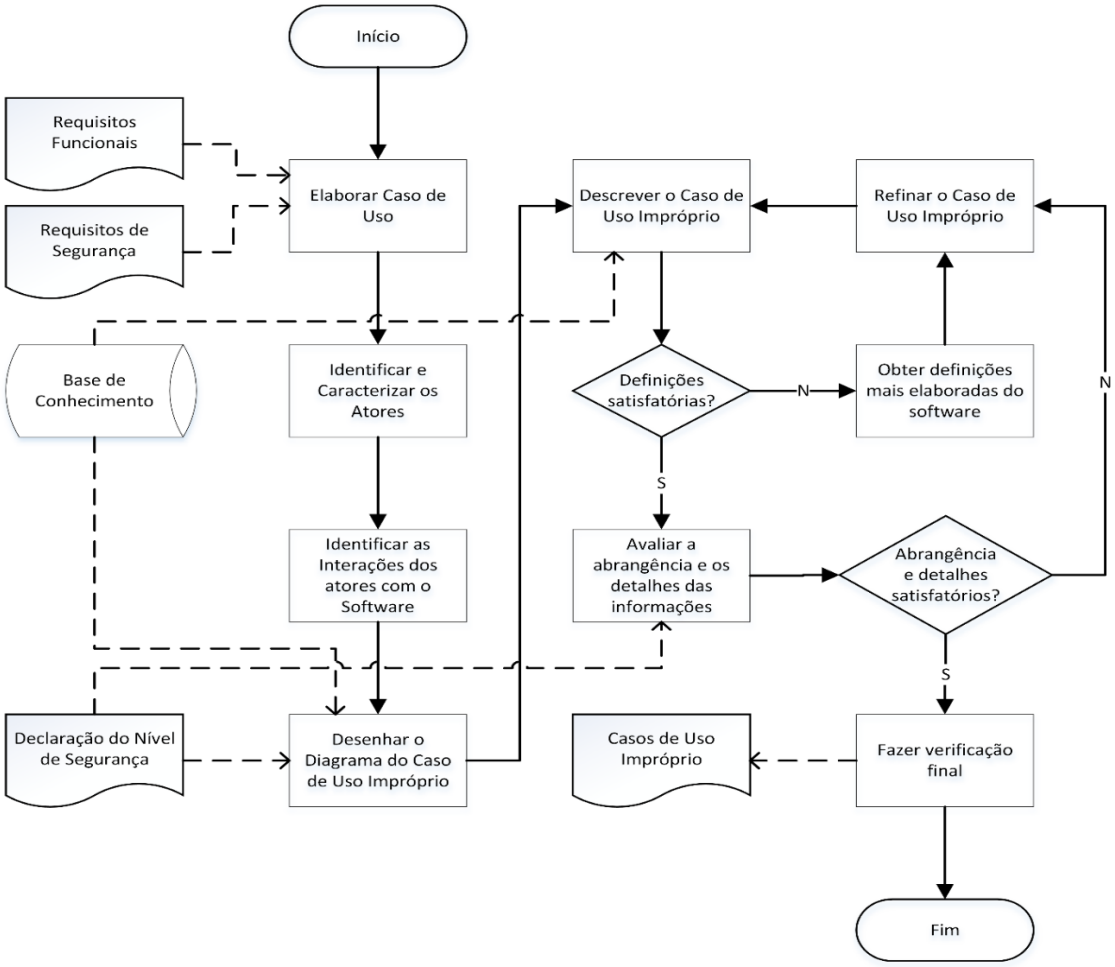


Figura 14 - Processo de elaboração de casos de uso impróprio.

Uma vez descritos os casos de uso impróprio, avalia-se a cobertura das possíveis e prováveis ocorrências, agrupando-se os casos de uso impróprio por similaridade, detalhando, simplificando ou desmembrando aqueles cuja complexidade ou extensão dificultam a análise ou a compreensão. Finalmente faz-se uma verificação de integridade e consistência, confrontando os casos de uso impróprio com os requisitos e com os casos de uso. Para os exemplos apresentados, considerando-se sua simplicidade, os dois casos de uso impróprio atendem aos critérios analisados, e passam a fazer parte da documentação do software para subsidiar as atividades seguintes do processo de desenvolvimento.

Ao executar a tarefa de produzir os casos de uso impróprio exercita-se a exploração e o detalhamento das atividades que podem resultar em danos à segurança da informação tratada pelo software. Isto conduz a atividade de desenvolvimento no sentido de caracterizar devidamente tais possibilidades e aprofundar o conhecimento acerca das mesmas, subsidiando assim a elaboração de contramedidas que eliminem ou atenuem tais ocorrências e, até mesmo no pior dos casos, avisem ao usuário do software acerca do ocorrido.

Apresentadas de maneira organizada e coerente, tais informações serão fundamentais para a seleção e implementação de contramedidas e da elaboração e execução de teste mais efetivos, bem como o monitoramento de tais ocorrências, de modo a reforçar a segurança da informação não somente no SDLC, mas por todo o ciclo de vida do software. Além disso, o processo de elaboração dos casos de uso impróprio converge para um dos objetivos da SDD, que é a aproximação das equipes de desenvolvimento com as equipes de segurança da informação das organizações. Para executar esta atividade é imprescindível a proximidade e o trabalho colaborativo.

4.5 Análise de risco

O modelo SDD propõe que a análise de riscos deva ir além da prática corrente, que a utiliza para embasar o planejamento do projeto de desenvolvimento de software, avaliando os riscos para o negócio em uma perspectiva financeira. Para o SDD a análise de riscos deve ser estendida às características do software em desenvolvimento e aos requisitos com o intuito de definir a prioridade no tratamento e a necessidade de tratamento para cada risco, que servirá como base para a tomada de decisão quanto a reduzir, reter, evitar ou repassar o risco.

No modelo SDD a análise de risco é uma atividade essencial no processo de garantia da segurança da informação. Para tanto, esta análise deve ser elaborada já no início do processo de desenvolvimento, em um nível genérico, após a especificação de requisitos, quando já é possível confrontar os requisitos com as ameaças e vulnerabilidades. Na sequência do processo de desenvolvimento é possível avaliar aspectos da arquitetura com o mesmo enfoque, com o intuito de ponderar os riscos.

É importante ressaltar que o SDD não tem por objetivo definir os critérios utilizados para a análise de riscos, uma vez que cada organização tem seu perfil de risco, que reflete sua sensibilidade ao risco, sendo umas mais avessas e outras mais tolerantes. E esta sensibilidade também pode variar a cada produto de software.

4.5.1 Métodos e enfoque

O modelo SDD não estabelece um método a ser empregado, os quais podem variar bastante entre si, desde aqueles baseados em modelos comerciais – como o STRIDE, da Microsoft – ou padrões como o OCTAVE, do SEI – *Software Engineering Institute*. Além disso, como abordado na Seção 3.3, há uma variedade de métodos, técnicas e ferramentas para a análise de risco originadas ou empregadas em diversas indústrias, tornando possível o uso daquela que for mais adequada ao tipo de software, às características do negócio, do projeto e das equipes, bem como dos próprios clientes ou usuários. O importante é que o pessoal envolvido com a atividade de análise de risco sinta-se confortável com o método, a técnica ou a ferramenta empregada, e conheça-os para que possa realizar a identificação dos riscos e as estimativas do modo mais acurado possível.

O enfoque da análise de risco também pode variar em função do propósito do software e das características dos usuários ou do cliente. Pode-se enfatizar o valor financeiro das perdas ocasionadas pelos riscos quando concretizados. Pode-se utilizar critérios numéricos, tabulando as ameaças, as probabilidades e os impactos das ocorrências. E pode-se utilizar uma base qualitativa, considerando o conhecimento e o registro histórico dos riscos e suas consequências.

Além desses enfoques, em um software desenvolvido para dispositivos móveis é necessário considerar os riscos nas diversas camadas que deverão prover serviços, tais como:

- A interface com o usuário;
- As capacidades do equipamento, como câmera, microfone, GPS;

- As funcionalidades do ambiente operacional, no caso o Sistema Operacional Android;
- As funcionalidades de comunicação, como a rede de telefonia (3G, 4G, etc.), WiFi, Bluetooth, SMS/MMS e NFC;
- Os serviços de banco de dados e acesso a dados;
- O ambiente computacional como um todo, considerando a capacidade de processamento, de armazenamento e até mesmo a fonte de energia – a bateria.

No que se refere a infraestrutura necessária para o desenvolvimento e operação do software, os seguintes aspectos devem ser considerados na avaliação dos riscos:

- O ambiente operacional, bibliotecas e hardware – e suas configurações para a operação do software;
- Os canais de comunicação – redes e infraestrutura utilizada;
- O ambiente de desenvolvimento – hardware e software – como um todo.

Na Seção C.2 do Apêndice C é apresentado um exemplo de Análise de Riscos com base no modelo discutido nesta seção.

4.5.2 Classificação e priorização do tratamento dos riscos

Uma vez identificados os riscos aos quais está sujeito o software, a Análise de Risco busca elaborar uma estimativa que possibilite a classificação dos riscos quanto ao impacto nos objetivos do software, a prioridade no tratamento, e a real necessidade de tratamento dos riscos. Este processo é denominado análise qualitativa dos riscos.

Para o estabelecimento das prioridades por meio da análise qualitativa dos riscos utiliza-se uma tabela – também referenciada como Matriz P-I – expressando os níveis de probabilidade X impacto, na qual são inseridos os itens de risco considerando estas características para cada um dos riscos apontados. Esta tabela apresenta de uma maneira rápida o panorama dos riscos e possibilita a avaliação da necessidade de providências e contramedidas. Um exemplo é apresentado na Tabela 4 a seguir, que associa à cada célula os itens de riscos relacionados nas tabelas 7, 8, 9 e 10[5], considerando os níveis citados de impacto e probabilidade de ocorrência:

[5] Estas tabelas são apresentadas na Seção C.2 do Apêndice C.

Tabela 4 – Matriz P-I: Priorização dos riscos.

		Probabilidade				
		Muito Baixa	Baixa	Média	Alta	Muito Alta
Impacto	Muito Alto				#16	
	Alto	#9	#2, #4, #5, #10	#11	#1, #3, #17	
	Médio	#8	#18, #19, #20	#7	#9, #15	
	Baixo		#15, #16	#6		
	Muito Baixo				#14	

A dificuldade com esta atividade da análise de risco é adequar o referencial para o estabelecimento dos níveis, ou seja, a definição da escala de probabilidade e de impacto do risco, e ajustá-la ao perfil de risco da organização. Em termos numéricos a probabilidade ocupa a faixa entre 0.0 (0% - certeza de que o risco não ocorrerá) e 1.0 (100% - certeza de que o risco ocorrerá). A avaliação da probabilidade torna-se complexa à medida em que julgamentos de especialistas são usados, normalmente quando não se pode contar com o auxílio de dados históricos.

Uma escala ordinal, representando valores de probabilidade relativa variando de muito baixa (ou muito pouco provável) a muito alta (ou quase certa), como a que foi usada neste caso, é a mais típica. Porém podem ser utilizadas probabilidades específicas usando uma escala numérica, em geral com valores padronizados, como por exemplo: 0.1(10% ou Muito Baixa); 0.3 (30% ou Baixa); 0.5 (50% ou Média); 0,7 (70% ou Alta) e 0.9 (90% ou Muito Alta).

A escala de impacto do risco reflete a gravidade do efeito ou consequências de uma ocorrência do risco nos objetivos do software. O impacto também pode ser expresso de modo ordinal ou cardinal. Escalas ordinais são simplesmente valores classificados por ordem, tal como a que foi utilizada neste caso: Muito alto, alto, médio, baixo e muito baixo. Escalas cardinais designam valores a estes impactos, que são normalmente lineares, como por exemplo: 0.1(10% ou Muito Baixo), 0.3 (30% ou Baixo), 0.5 (50% ou Médio), 0,7 (70% ou Alto) e 0.9 (90% ou Muito Alto).

As escalas cardinais podem também ser não-lineares, como por ex., 0.05, 0.1, 0.2, 0.4 e 0.8, adequando-se ao perfil de risco da organização ou do cliente do software, com o propósito de evitar ou correr riscos de alto impacto. A intenção de ambas as abordagens é designar um valor relativo ao impacto causado se o risco em questão ocorrer. Escalas bem definidas, sejam ordinais ou cardinais, são desenvolvidas usando definições combinadas com a organização. Estas definições melhoram a qualidade dos dados e torna o processo mais repetitivo.

Para auxiliar na utilização da escala e na classificação dos riscos deve-se utilizar os critérios mais objetivos possíveis. No caso da probabilidade foram utilizadas as premissas do programa de ônibus espaciais estabelecidas pela NASA (2001) para a escala de impacto, e considerou-se um histórico de ocorrências semelhantes em determinado período de tempo, estabelecido como um dia. Aplicou-se então um fator ao período de tempo, como apresentado na Tabela 5.

Tabela 5 - Escala de Probabilidade em função do tempo.

	Probabilidade				
	Muito Baixa	**Baixa**	**Média**	**Alta**	**Muito Alta**
Fator	1/1000	1/100	1/10	1/3	2/3
Ocorrências	Uma a cada 1000 períodos	Uma a cada 100 períodos	Uma a cada 10 períodos	Uma a cada 3 períodos	Duas a cada 3 períodos

Já para o impacto deve-se considerar os aspectos técnicos como a extensão do dano ou área afetada, o comprometimento dos objetivos (no caso dos requisitos) do software, a redução da capacidade operacional ou das características da segurança da informação atingidas – confidencialidade, integridade, disponibilidade, etc. Considera-se também o tempo de reparação ou normalização e o custo desta reparação ou restabelecimento da situação. No caso em questão também foram utilizadas as premissas do programa de ônibus espaciais estabelecidas pela NASA (2001) para a escala de impacto, que considera o impacto do ponto de vista da complexidade técnica, do tempo e do custo, conforme mostrado na Tabela 6 a seguir.

Tabela 6 - Escala de Impacto com base em critérios técnicos, de tempo e custo. Adaptado de NASA (2001).

	Impacto				
	Muito Baixo	**Baixo**	**Médio**	**Alto**	**Muito Alto**
Técnico	Mínimo ou nenhum	Redução moderada: é possível prosseguir	Redução sensível: possibilidade do uso de alternativas	Grande redução: necessidade do uso de alternativas	Inaceitável: não existem alternativas
Tempo	Mínimo ou nenhum	Requer atividades adicionais sem afetar a agenda / calendário	Afeta a execução de atividades da agenda / calendário em alguns dias	Afeta a execução de atividades da agenda / calendário em alguns meses	Impossibilita a execução de atividades
Custo	Menor que $100	Entre $100 e $1000	Entre $1000 e $10K	Entre $10K e $50K	Acima de $50K

4.5.3 Tratamento e acompanhamento dos riscos

Uma vez executada a análise dos riscos é necessário apreciar ou caracterizar os riscos, isto é, avaliar os riscos analisados. O objetivo desta avaliação é decidir o que fazer perante os riscos identificados, estabelecendo as prioridades e analisando a efetividade e a real necessidade do tratamento dos riscos analisados. Para um enfoque financeiro pode ser necessário proceder a uma análise quantitativa do risco e uma avaliação do ROI – *Return of Investment* - Retorno do Investimento, que vem a ser um critério importante para a definição do tratamento a ser dispensado ao risco, processo denominado minimização ou mitigação do risco. As opções do tratamento do risco são definidas em função da análise de risco (prioridade), custo estimado da implementação e resultados esperados. As opções para o tratamento dos riscos são as seguintes:

- Reduzir: criação de mecanismos que reduzam a probabilidade ou o impacto do risco;
- Reter ou Evitar: utilização de medidas que impeçam a ocorrência do risco pela eliminação de vulnerabilidades ou tratamento contra as ameaças, como a modificação ou substituição dos elementos que apresentam vulnerabilidades ou o uso de alternativas mais seguras;
- Transferir: indicação de terceiros para adotar as providências para a mitigação do risco ou a administração das consequências.
- Aceitar: mesmo aplicando um tratamento aos riscos é improvável que se consiga eliminá-los totalmente. Sempre haverá um risco residual, por menor que seja, que fará parte do processo de aceitação do risco, dada a sua inexorabilidade ou a

inviabilidade técnica ou financeira do tratamento. Esta aceitação deve ser comunicada e formalizada nos documentos do projeto do software para tornar-se de conhecimento de todos, inclusive dos usuários e clientes.

Finalmente são necessários o monitoramento e a revisão da análise de riscos nas fases posteriores, notadamente após a implementação do tratamento dos riscos, durante as fases de construção e teste do software. Isto deve-se ao fato de que os riscos não são estáticos, uma vez que as ameaças e vulnerabilidades, as probabilidades e o impacto podem mudar – e mudam – no decorrer do SDLC.

Durante o processo de desenvolvimento podem mudanças internas e externas da organização, da legislação, das necessidades de segurança e dos próprios requisitos do software que vão alterar os cenários de risco. Também é necessário confrontar os riscos com as medidas adotadas para o tratamento deles, a fim de aferir a efetividade das mesmas. Por isso é necessário revisitar e revisar a análise de riscos à cada etapa, de forma recorrente, e também atualizar as informações a respeito de novos riscos identificados e tratados. Estas informações devem realimentar a base de conhecimentos e refletir nas demais etapas do processo de desenvolvimento, podendo mesmo representar a revisão dos procedimentos adotados e até mesmo em retrabalho, caso seja necessário intervir em partes já executadas dos trabalhos.

4.6 Máquinas de ataque

Um aspecto relativo à segurança da informação que requer atenção é o teste o software, ainda no SDLC. Devido ao fato de não haver, de um modo geral, um ferramental específico para testar os requisitos de segurança, os testes de segurança são executados após a entrega, às vezes já no ambiente de produção ou operação do software, e com ênfase em ataques de *malwares*, por meio de teste de invasão. Esta abordagem, apesar de eficaz para os propósitos aos quais se destina, carrega duas deficiências:

1) Como tratado neste trabalho, os problemas de segurança da informação não se restringem aos *malwares* e ataques externos ao software, especialmente no ambiente de dispositivos móveis, cuja complexidade e diversidade de elementos e fatores sujeitos às ameaças e agentes de vulnerabilidades foram abordados no Capítulo 2;
2) Teste de invasão ou teste de segurança são, de um modo em geral, bastante voltados para vulnerabilidades do sistema operacional e da infraestrutura, como destacado na

Seção 3.5, poucas vezes explorando os requisitos de segurança da informação do software propriamente dito.

O modelo SDD propõe enfrentar estas deficiências com a construção de máquinas de ataque que possam ser empregadas para testar a segurança do software muito além das ameaças de *malwares*, e que possam atuar sobre os requisitos de segurança da informação específicos de cada software.

Máquinas de ataque são programas, componentes ou trechos de código construídos para simular a ação de *malwares*, invasores, usuários mal informados, mal-intencionados ou as falhas, erros e faltas que possam impedir o software de atingir seus objetivos. As principais finalidades do projeto e da construção das máquinas de ataque são:

- Validar os danos previstos e descritos nos casos de uso impróprio;
- Simular a ocorrência dos riscos elencados na análise de riscos;
- Comprovar a efetividade das contramedidas propostas e adotadas;
- Automatizar as atividades de teste o tanto quanto possível, seja pela integração com as ferramentas de teste automatizado seja pela possibilidade de execução programada e automática.

Além disso, as máquinas de ataque são essenciais ao processo de teste pois, sendo criadas em função de riscos específicos, possibilitam a execução de testes mais objetivos – em contraponto aos testes genéricos – reduzindo o esforço e aumentando a efetividade do teste, com a consequente redução do risco de que o software venha a apresentar problemas decorrentes de ataques quando estiver em produção, em especial aqueles decorrentes de problemas já conhecidos.

4.6.1 Modelagem de máquinas de ataque

Com uma abordagem semelhante à modelagem de ameaças, a modelagem das máquinas de ataque propõe o aproveitamento máximo de características comuns das ameaças para propiciar a construção de elementos que explorem as vulnerabilidades pontualmente. E assim como a modelagem de ameaças, a modelagem das máquinas de ataque tem como insumos básicos:

- O estudo e a classificação das vulnerabilidades do ambiente operacional, da infraestrutura e suas características funcionais, como tratado no Capítulo 2;

- A classificação das ameaças e a base de conhecimento acerca das mesmas, incluindo características operacionais e até mesmo código, como tratado na Seção 2.5 e no Apêndice A;
- Os casos de uso impróprio desenvolvidos para as ameaças devidamente classificadas, como exposto nas Seções 4.4 e no Apêndice C;
- A análise de riscos, principalmente no que se refere à priorização do tratamento dos riscos, com abordado nas Seções 4.5 e no exemplo do Apêndice C;

Um facilitador adicional da modelagem de máquinas de ataque é a sua vinculação e dependência com o ambiente e a infraestrutura do software, o que significa que o mesmo time de projeto do software está, a princípio, habilitado a projetar e construir as máquinas de ataque.

Além disso, é comum o interesse dos desenvolvedores de software pelas atividades dos *bad guys*, os desenvolvedores de *malwares*. O fato de existirem outros profissionais de sua classe que possuem habilidades para identificar e explorar vulnerabilidades desperta a curiosidade e o interesse, e pode ser utilizado como um motivador, um desafio para a equipe de desenvolvimento empregar os seus melhores esforços para provar sua capacidade de operar em condições de igualdade com aqueles. Ou seja, o projeto e a construção das máquinas de ataque são uma ótima forma dos desenvolvedores praticarem a segurança do ponto de vista dos *bad guys*, desafiando-os a se mostrarem mais competentes e preparados que aqueles.

Outra característica desta proposta do modelo SDD é justamente a necessidade do trabalho colaborativo e interdependente entre as equipes de desenvolvimento – que detém a expertise necessária para a construção das máquinas de ataque, e as equipes de segurança da informação, que detém o conhecimento das ameaças e vulnerabilidades, resultando em sinergia e na eliminação do *gap* relatado por McGraw (2005).

No modelo de SDD a etapa de especificação do software inclui a criação de casos de uso impróprio específicos ou genéricos, confrontando riscos e vulnerabilidades e contemplando ameaças identificadas e posteriormente caracterizadas como relevantes pela análise de riscos. No caso do projeto de software utilizado como exemplo, estes casos de uso impróprio embasam a modelagem de máquinas de ataque, cujo processo de elaboração é detalhado na Seção C.3 do Apêndice C, descrevendo e apresentando os modelos das respectivas máquinas de ataque.

4.6.2 Aplicação de máquinas de ataques nos testes de segurança

No modelo SDD, uma vez projetadas e construídas as máquinas de ataques modeladas conforme descrito anteriormente, os testes específicos para a segurança do software contemplam o uso destas máquinas de ataque para a execução dos testes, com o intuito de aferir a eficácia dos mecanismos de defesa e as contramedidas de segurança adotadas. Para tanto os casos de teste devem contemplar as operações específicas voltadas para o uso destas máquinas, com base nos casos de uso impróprio e na análise de riscos.

4.7 Testes de segurança

O modelo SDD visa a adequação e a efetividade do teste de segurança da informação buscando estabelecer um ponto de equilíbrio entre a flexibilidade, facilidade de implementação, recursos necessários e custos dos processos de teste abordados. Isto implica em estabelecer o mais cedo possível os critérios viáveis de avaliação, usando o conhecimento das técnicas e métodos de testes voltados à segurança da informação como forma de prover aplicações e serviços seguros.

O início das atividades de teste deve ser o mais antecipado possível dentro do SDLC, independente do modelo e das técnicas empregadas. Os requisitos de segurança, que serão a base para a definição dos casos de teste, devem fazer parte do processo formal e serem definidos juntamente com o processo de especificação funcional do software a ser desenvolvido. A partir daí, a produção e o refinamento de casos de teste deve ter sequência a cada iteração, fase ou ciclo do SDLC, de modo que ao chegar à etapa de testes de homologação ou aceitação pelo usuário tais requisitos tenham sido atendidos satisfatoriamente e a segurança implícita – que faz parte da expectativa do usuário – seja efetiva e comprovada.

Como já visto, para que isso aconteça o SDD estabelece que é necessário que se inicie o SDLC analisando o risco inerente ao uso da informação para a qual o software está sendo projetado. Desta forma são estabelecidos os valores de cada informação manipulada ou produzida pelo software, e a prioridade de tratamento da segurança da informação de acordo com esse valor. Em seguida deverá ser analisada a exposição dessas informações ao risco, isto é, as vulnerabilidades e as ameaças que possam explorar essas vulnerabilidades. Para isso é necessário que o desenvolvedor raciocine de modo diferente, assumindo o papel de atacante, com o intuito de explorar as vulnerabilidades do software, produzindo os casos de uso

impróprio, cenários nos quais registrará suas ações para violar a segurança da informação tendo como premissa as vulnerabilidades do software e as ameaças conhecidas.

Os casos de uso impróprio serão utilizados para a modelagem dos procedimentos de resposta aos ataques – as contramedidas - a serem implementados para eliminar ou reduzir as vulnerabilidades a níveis aceitáveis, conforme definido na análise de risco. E também possibilitarão a definição dos casos de teste para a experimentação da efetividade das contramedidas estabelecidas, e a especificação e construção das máquinas de ataque, que serão utilizadas para os testes de segurança e para os testes de invasão.

Os principais problemas de uma abordagem precoce do teste em geral – e do teste de segurança da informação em específico – no SDLC decorrem da pressão exercida pelos prazos e custos. Além disso, o baixo nível de automação de tarefas de teste, a geração e manipulação de massa de teste e a necessidade de manter constante atualização sobre as técnicas de ataques e as vulnerabilidades exploradas representam um esforço adicional significativo e que coloca o processo de teste em evidência quando há a necessidade de redução de esforço, de custos ou de prazo.

A abordagem tradicional, na qual os procedimentos de teste de segurança são tratados em sua plenitude somente após os ciclos de desenvolvimento, fomenta uma prática pouco recomendável de recompor o cronograma e efetuar ajustes de esforço ou de custos reduzindo-se as atividades de teste, e assim comprometendo a qualidade do software. Além disso, uma abordagem tardia impede a implementação de um processo cíclico de teste, interagindo com o desenvolvimento e com a análise de riscos constantemente atualizada e revista, o que só é possível conseguir com uma abordagem proativa e assertiva da questão da segurança da informação no processo de desenvolvimento de software.

Para enfrentar estas situações o modelo SDD define a elaboração do plano de teste na sequência da análise de riscos e a elaboração do teste de segurança baseados em riscos ainda durante a concepção do software, na fase de análise e projeto, e a construção das máquinas de ataque em conjunto com a construção do software. Com isto o ferramental básico para os testes de segurança é disponibilizado a tempo, reduzindo a pressão na etapa de testes propriamente dita.

Outro aspecto do modelo SDD que visa reforçar a efetividade do teste é a automação, neste caso por meio da utilização das máquinas de ataque que, além de fazerem parte do universo conhecido pela equipe de desenvolvimento por terem sido por ela desenvolvidas,

também devem ter sido projetadas para a integração com ferramentas de automação de teste, como já salientado.

E finalmente, a base de conhecimento que faz parte da aplicação do modelo SDD é um importante referencial tanto para a elaboração do plano de teste e do teste de segurança em si quanto para a execução deste teste, uma vez que retrata o comportamento, os objetivos e principalmente o funcionamento das ameaças, o que permite reproduzir os ataques e os problemas de forma real.

4.8 Segurança da operação

No modelo SDD, após a implantação do software é necessário manter a avaliação da segurança da informação do software. Além da função de suporte operacional normalmente prestado, a continuidade da operação tem por objetivo realimentar a base de conhecimento por meio do registro de ocorrências de segurança da informação pelas equipes de suporte. Este registro servirá para avaliar o comportamento do produto e a efetividade das contramedidas, além de sinalizar quando da ocorrência de novas ameaças ou mudanças no cenário de operação do software

Também deve ser providenciado o registro de ocorrências por meio do próprio software, desde que consentido pelo usuário, encaminhando para a equipe de desenvolvimento e da segurança da informação um relato das ocorrências e exceções, de forma a permitir a análise da ocorrência, o tratamento por meio da aplicação de correções e a atualização da base de conhecimento. Esta prática já é adotada por grande parte dos fornecedores de software comercial e contribui para o melhoramento da qualidade dos produtos.

Outra forma de *feedback* de grande importância é o relato de problemas por parte dos próprios usuários por meio de canais de atendimento como SACs, e-mail e redes sociais. Em especial é possível identificar particularidades das ocorrências e também surtos epidêmicos típicos de novas ameaças.

O modelo SDD inclui na segurança da operação o trabalho voltado ao recebimento, análise, refinamento e classificação das informações relativas à segurança da informação e o respectivo registro na base de conhecimento para realimentar o modelo no SDLC, possibilitando o melhoramento contínuo da segurança da informação no software. Uma possibilidade de promover este *feedback* é a inclusão de uma funcionalidade de tratamento de

erros que permita o registro da opinião e das considerações do usuário de forma anônima, seja em formulário específico na Internet, e-mail ou mensagem SMS/MMS.

4.9 Estudo de Caso

Com o intuito de apresentar o uso do modelo passo a passo e prover mais exemplos dos artefatos e atividades do modelo, é apresentado um estudo de caso[6] utilizando o modelo SDD em um projeto piloto de software. O software proposto, denominado µCRM, é uma aplicação de usuário para *smart phones* com Android que tem por objetivo complementar a função "Agenda" existente nesse ambiente, classificando e expandindo as informações da agenda, das chamadas telefônicas e mensagens – de texto ou multimídia – trocadas com números de telefone nela constantes.

O objetivo principal do µCRM é auxiliar os profissionais – principalmente os da força de vendas das organizações - na manutenção dos registros dos contatos realizados, por meio do telefone e de mensagens, com os clientes.

A aplicação encarrega-se de verificar a existência de chamadas e mensagens e, quando associadas a números previamente identificados, fazer o registro em um banco de dados, permitindo a inserção de informações de diversos tipos – texto, áudio, imagens e vídeo – que serão associados ao evento e à agenda, podendo ser recuperados posteriormente. Com isto o usuário poderá manter um registro de interação com os contatos de sua agenda aprimorando o relacionamento. Estas informações podem ser consultadas e exportadas em formato de documento, base de dados ou relatórios, ou enviadas por e-mail, quando solicitadas.

Mesmo considerando-se que o protótipo de sistema utilizado para o estudo de caso seja bastante simples, há desafios na sua construção que reforçam a necessidade da aplicação do modelo SDD. O simples fato de tratar das informações de cunho pessoal que são utilizadas e armazenadas em um dispositivo móvel, ao qual o acesso é facilitado, de uma maneira em geral, e que está sujeito a danos, perda e roubo, implica na necessidade de um tratamento especial. E isto mesmo para uma aplicação bem simples como a apresentada neste estudo de caso.

Considerando este cenário é possível inferir que a aplicação do modelo SDD ao processo de desenvolvimento do sistema µCRM pode viabilizar a abordagem holística que se pretende, sem, entretanto, interferir demasiadamente no processo de desenvolvimento e sem

[6] O estudo de caso é apresentado em toda sua extensão no Apêndice D.

requerer mudanças expressivas, quer seja no modelo quer seja no ferramental a ser utilizado no processo de desenvolvimento.

Além disso a execução das atividades que compõem o modelo implica em um incremento no conhecimento de áreas correlatas, quer seja, para os profissionais de segurança da informação um avanço sobre o SDLC e os modelos de desenvolvimento de software, e para os desenvolvedores um aprofundamento em métodos e técnicas que estão no domínio da segurança da informação.

Esta convergência de conhecimentos e de atuação dos profissionais é muito bem vista e pode representar um avanço significativo no sentido de reforçar a abordagem de segurança da informação no processo de desenvolvimento de software para dispositivos móveis, proposta basilar deste trabalho.

4.10 Considerações do Capítulo

Este capítulo apresentou a proposta do modelo SDD para incrementar a segurança da informação no SDLC e, por conseguinte, a segurança da informação em todo o ciclo de vida do software. Foram apresentados os principais aspectos do modelo SDD e a expectativa de resultados advinda de seu uso, assim como exemplos e *templates* do modelo SDD para serem utilizados como referência.

O modelo complementar para aprimorar a segurança da informação no SDLC, o SDD – *Security Driven Developing*, é uma resposta à necessidade de incrementar o nível de segurança da informação no software para ambiente de dispositivos móveis. O SDD decorre da pesquisa de soluções e propostas que reforcem a segurança da informação no SDLC, das normas, das boas práticas e da observação de situações recorrentes na indústria de software que têm um impacto direto na segurança da informação.

Um dos objetivos do modelo SDD é reduzir o distanciamento entre as equipes de segurança da informação e as de desenvolvimento de software, pois pode haver uma separação e até mesmo competição e antagonismo entre estas equipes. Com o estudo de material e pesquisas realizadas no presente trabalho, é possível inferir que há lacunas no processo de desenvolvimento de software que decorrem desta situação, trazendo como resultado impactos relevantes na segurança da informação das aplicações desenvolvidas. Estas lacunas compreendem desde questões ligadas à formação profissional até as próprias políticas de

segurança da informação das organizações – em especial as voltadas para o desenvolvimento de produtos de software.

Outro ponto no qual o SDD atua fortemente é o desconhecimento de aspectos fundamentais da segurança de informação por parte dos desenvolvedores - que de uma forma em geral entendem como ameaça o conjunto de técnicas *hacker* para quebra da segurança, ou então uma lista de *malwares*. Este desconhecimento faz com que a segurança da informação seja reduzida a um pequeno conjunto de boas práticas e à delegação de responsabilidades da segurança da informação para os usuários, para os ambientes operacionais, softwares antivírus e outras ferramentas. Esta atitude deixa o produto de software que produzem bastante vulnerável e a mercê de ameaças, dependentes unicamente da proteção de tais mecanismos de segurança de terceiros.

O crescimento do uso dos dispositivos móveis trouxe um fator adicional de preocupação para esta situação, uma vez que, dado o mercado apresentado aos desenvolvedores, tornou-se atrativo o desenvolvimento de aplicações as mais diversas para este ambiente. A facilidade para criar e disponibilizar as aplicações, que são prontamente adquiridas por milhares de usuários, embute o risco de processos de desenvolvimento apressados e sem condições mínimas de gestão da segurança da informação, quer do processo de desenvolvimento quer das informações manipuladas pelo software em ambiente de execução.

Os profissionais de segurança, por sua vez, orientados a prover segurança da infraestrutura, pouco participam do processo de desenvolvimento de software, interagindo com este na maioria das vezes somente a partir da etapa de teste, quando o software já está no caminho da disponibilização para o usuário. Ou seja, tarde demais.

Outra situação verificada do decorrer das pesquisas realizadas e na interação com profissionais de diversos projetos em organizações voltadas para a produção de software é que há uma segmentação no tratamento da segurança da informação, com uma concentração das responsabilidades nas fases de construção e teste. E no geral este tratamento é composto fundamentalmente de boas práticas e técnicas de segurança reativas, com pouca ou quase nenhuma participação dos interessados e menor ainda interação com as áreas de negócio da organização na elaboração de contramedidas. Há pouca inovação no modelo de segurança das aplicações e nas recomendações aos usuários, e o resultado imediato é uma recorrência de problemas amplamente conhecidos, cujas soluções são de domínio público e – na maioria das vezes – de fácil adoção.

No próximo capítulo são apresentadas as conclusões deste trabalho de pesquisa e também uma projeção para a continuidade do trabalho, com o intuito de consolidar o modelo SDD como uma opção viável para a construção de software confiável e seguro para os dispositivos móveis.

Capítulo 5 - Conclusão

5.1 Conclusão

Neste trabalho demonstrou-se a necessidade de abordar a segurança da informação em todo o SDLC de dispositivos móveis, perante a crescente utilização de softwares dos mais diversos tipos e das características das informações disponíveis nestes dispositivos. O que observou-se até o presente momento é que, embora haja uma preocupação generalizada com a segurança da informação, no geral os esforços para garanti-la são sentidos nas iniciativas da indústria de equipamentos, do sistema operacional e das operadoras de telefonia móvel. Os cuidados com o desenvolvimento de software são restritos aos aspectos funcionais e à qualidade em geral, não considerando a confiabilidade e a segurança do software.

No desenvolvimento deste trabalho observou-se que há uma considerável parcela dos problemas de segurança da informação que pode ser tratada com o uso de processos, normas e padrões já existentes, muitos dos quais já são empregados, porém de forma limitada, como a análise de riscos e o teste de segurança. Isto faz com que as questões relativas à segurança fiquem à margem do SDLC ou relegadas aos cuidados do próprio usuário, ao sistema operacional ou à uma ferramenta de defesa, como um antivírus. E a convivência com a insegurança traz a desconfiança e o receio do usuário, ao mesmo tempo em que amplia o raio de ação dos mal-intencionados e criminosos.

Como forma de corrigir este problema, este trabalho apresentou uma revisão de diversos modelos já conhecidos, acrescentando características, informações e técnicas que, utilizadas em conjunto e com o mesmo propósito – essência do modelo SDD - permitem o tratamento da segurança da informação desde as primeiras atividades do SDLC, e assim contribuem significativamente para aprimorar a segurança da informação do software em desenvolvimento.

Devido à abordagem holística, contemplando todo o processo de desenvolvimento e além dele, o modelo SDD possibilita não só o planejamento do software com o enfoque na segurança da informação, como também a construção, verificação, validação e mesmo na utilização, trazendo para o projeto do software uma maior responsabilidade com os aspectos de segurança e também a apresentando aos usuários, de forma que haja a percepção da

segurança da informação. Evidentemente existirão necessidades diferentes de segurança em função do perfil dos usuários, do tipo de negócio envolvido, da cultura, dos recursos computacionais e outros fatores. E é por isso que o modelo SDD lança mão de uma avaliação prévia para direcionar todo o esforço para as ações mais apropriadas e assim dar objetividade ao processo.

Começando pela identificação das vulnerabilidades e ameaças da forma mais ampla e detalhada possível, passando pela declaração do nível de segurança requerido, requisitos de segurança, elaboração dos casos de uso impróprio, análise de riscos, teste de segurança baseados em riscos e com uso de máquinas de ataque, e realimentando o processo com informações de segurança colhidas da operação, o modelo SDD apresenta um amplo conjunto de elementos para reforçar a segurança da informação no SDLC e no uso do software, permitindo aos desenvolvedores atuar de forma assertiva e proativa perante os riscos e as ameaças.

Deste ponto de vista é possível inferir que o uso do modelo proposto neste trabalho no desenvolvimento de aplicações para dispositivos móveis certamente trará resultados positivos quanto à confiabilidade e à qualidade do software, e também à percepção de segurança por parte dos usuários destes softwares. Softwares desenvolvidos com o apoio do modelo SDD serão, indubitavelmente, melhor preparados para enfrentar as ameaças e superar as vulnerabilidades do ambiente, tornando a sua utilização mais segura e melhor preservando a segurança da informação de seus usuários.

Embora a maioria das aplicações para dispositivos móveis não manipulem diretamente informações que necessitem de cuidado quanto à segurança, a forma como estas informações são mantidas e acessadas pelas aplicações faz com que deva haver cuidado adicional com o armazenamento e o trânsito destas informações. Em função disto, para a maioria da informação mantida nestes dispositivos, os aspectos básicos da segurança da informação - confidencialidade, integridade e disponibilidade – não podem ser negligenciados por qualquer tipo de software, sob pena de comprometer todo o ambiente do dispositivo.

Um aspecto relevante a ser considerado quanto aos dispositivos móveis é justamente o seu uso no ambiente corporativo, ou como extensão do ambiente computacional das organizações no caso do BYOD. A proteção do ambiente em geral deve levar em

consideração a capacidade dos dispositivos móveis e a segurança fornecida pelo sistema operacional e pelos softwares neles instalados, razão pela qual as organizações também devem primar não só pelo desenvolvimento das suas próprias aplicações confiáveis neste ambiente, mas da possibilidade de certificar-se da confiabilidade das demais aplicações no mesmo ambiente, o que pode ser providenciado pela aplicação do modelo SDD.

Conhecendo os problemas e riscos inerentes ao uso dos dispositivos móveis e do sistema operacional, as organizações que optam por desenvolver softwares para este ambiente aceitam os riscos e devem encarar as vulnerabilidades e as ameaças que a plataforma apresenta. O modelo SDD vem de encontro à necessidade de mitigar estes riscos no processo de desenvolvimento do software, possibilitando uma análise de risco efetiva e o enfrentamento das ameaças, incluindo as correções de falhas antes que se tornem problemas de maior extensão.

O uso do modelo do SDD visa justamente reduzir a possibilidade de que problemas relativos à segurança da informação sejam identificados após a disponibilização do software para uso, ou até mesmo que este software venha acrescentar vulnerabilidades ao ambiente do dispositivo móvel. Em síntese, o modelo integra um conjunto de boas práticas que trata os possíveis problemas – principalmente os mais típicos e os já conhecidos - de forma proativa, evitando que sejam detectados tardiamente quando o software já está em uso ou até mesmo por um agente mal-intencionado: o que se pretende é que a equipe de desenvolvimento faça estas descobertas ainda durante o SDLC e o mais precocemente possível, enquanto pode-se adotar providências e correções com o menor impacto possível.

A partir do momento em que o desenvolvimento dos softwares passar a adotar o modelo SDD haverá uma evolução na maturidade, e o cuidado para que o modelo SDD fosse algo simples tem o objetivo de torná-lo corriqueiro e de aplicação regular, sendo incorporado ao modelo de desenvolvimento adotado pela organização ou pelos projetos sem a necessidade de grandes adaptações ou nova capacitação das equipes. Isto gera um ciclo virtuoso que reverbera na cultura organizacional, e consequentemente impõem a necessidade de elevar a segurança da informação ao patamar de excelência que é requerido atualmente pelos usuários.

É certo que questões de segurança somente podem ser devidamente tratada com o comprometimento de todos, havendo a consciência coletiva dos males resultantes das falhas

– mesmo as menores. E o estabelecimento de uma cultura organizacional voltada para a segurança da informação, aliada a atuação de equipes capacitadas nas práticas propostas pelo modelo apresentado neste trabalho podem gerar sinergia e refletir na construção de um ambiente que produza software confiável e seguro, evoluindo de modo recorrente e antecipando-se aos problemas ao invés de reagir a eles, atingindo o patamar de melhoria contínua tão almejada pelas organizações.

Em decorrência de tudo isto é pertinente considerar que o modelo SDD, resultante deste trabalho, é adequado para a aplicação no SDLC independentemente da natureza do software, da maturidade da organização ou do modelo de desenvolvimento empregado. Apesar de não ter sido possível ainda determinar o impacto da aplicação do modelo SDD nos custos de desenvolvimento, a concepção simplificada do modelo e o uso de técnicas e práticas já conhecidas reforça a aderência e diminui significativamente o impacto nos custos. Entretanto é plausível considerar variações deste custo em função do modelo de desenvolvimento adotado, da maturidade das equipes envolvidas e dos próprios processos da organização, da complexidade do software, entre outros fatores.

Desta forma, este trabalho apresenta como contribuições:

1. Uma proposta viável para o enfoque da segurança da informação em todo o SDLC, aderente a diversos modelos utilizados no desenvolvimento de software para dispositivos móveis;
2. Uma proposta para a construção e a manutenção de uma base de conhecimento sobre ameaças e vulnerabilidades em dispositivos móveis;
3. Uma fundamentação para o estudo e catalogação de faltas, erros e falhas de dispositivos móveis, que permite compor uma sólida base de conhecimento a respeito;
4. Um modelo para o estudo, identificação e classificação de *malwares* voltados para os dispositivos móveis;
5. Um modelo para o projeto e a construção de máquinas de ataque para a efetivação de riscos identificados e a validação das contramedidas e proteções destinadas a reforçar segurança da informação nos dispositivos móveis;

6. A proposta de teste voltado para a segurança com o uso de máquinas de ataque, possibilitando um maior grau de automação e controle e tornando o teste mais objetivo e efetivo.

Finalizando, a aplicação do modelo no estudo de caso permitiu concluir que é possível obter bons resultados na identificação de problemas de segurança da informação durante o ciclo de desenvolvimento de software e tratar adequadamente estes problemas no momento mais oportuno, atingindo o objetivo proposto de aprimorar a segurança da informação no SDLC de dispositivos móveis.

5.2 Trabalhos futuros

Como continuidade para o presente trabalho considera-se a complementação do modelo e a extensão das avaliações por meio da aplicação em casos reais de desenvolvimento. Para a complementação é possível o desenvolvimento de um conjunto de ferramentas – o *SDD Toolkit* – que ampare a aplicação do modelo e permita a integração com outras ferramentas e a rastreabilidade das informações utilizadas e produzidas pela aplicação do modelo. Também é necessária a elaboração de um manual com *templates* para a implantação e para a utilização do modelo SDD.

A criação de um portal na Internet para a divulgação e discussão do modelo, interação com a comunidade de desenvolvedores, de profissionais de segurança e a comunidade acadêmica também faz parte da estratégia para a maturidade do modelo. Também é necessário o desenvolvimento de técnicas e métricas de aferição da efetividade do modelo e a universalização do modelo para a aplicação em outras plataformas e ambientes de computação, além dos dispositivos móveis.

Todas estas iniciativas representam grandes desafios, porém amplamente justificáveis em vista da busca contínua pelo aprimoramento da segurança da informação dos softwares em geral.

Referências

ABRAHAMSSON, P., SINIAALTO, M. *Does Test Driven Development Improve the Program Code? Alarming Results from a Comparative Case Study*. Proceedings of Second IFIP TC 2 Central and Europe Conference on Software Engineering Techniques, Poznan-Poland, October, 2007:143-156.

ADIBI, S. *Comparative mobile platforms security solutions*. 2014 IEEE 27th Canadian Conference on Electrical and Computer Engineering (CCECE).Toronto, Canada, May, 2014: 1-4.

ANTONIOL, G. *Search Based Software Testing for Software Security: Breaking Code to Make it Safer*. IEEE International Conference on Software Testing Verification and Validation Workshops. IEEE Computer Society, 2009:88-100.

AOYAMA, M. *New Age of Software Development: How Component-Based Software Engineering Changes the Way of Software Development?* ICSE'98 – International Workshop of Component-Based Software Engineering, 1998:1-4.

ARP 4754. *Certification considerations for highly-integrated or complex aircraft systems. SAE Systems Integration Requirements Task Group AS-IC*. 1994.

AVIZIENIS, A., LAPRIE, J. C., RANDELL, B., and LANDWEHR, C. *Basic Concepts and Taxonomy of Dependable and Secure Computing*. IEEE Transactions on Dependable and Secure Computing Vol.1 Nº1, 2004.

BRAGA, A. M., NASCIMENTO, E. N., PALMA, L. R., e ROSA, R. P. *MC2: Introdução à Segurança de Dispositivos Móveis Modernos - Um Estudo de Caso em Android*. Disponível em SBSeg'12: http://dainf.ct.utfpr.edu.br/~maziero/lib/exe/fetch.php/ceseg:2012-sbseg-mc2.pdf. Acesso em 07 de Fevereiro de 2013.

CHANDRAMOHAN, M., KUAN, H. B. *Detection of Mobile Malware in the Wild*. IEEE Computer Magazine, Vol. 45, Issue 9, November 2012:65-71.

CHANDRAMOULI, R. e BLACKBURN M. *Automated Testing of Security Functions using a combined Model & Interface driven Approach*. Proceedings of the 37th Hawaii International Conference on System Sciences. IEEE, 2004:1-10.

COCKBURN, A. *Crystal clear: a human-powered methodology for small teams*. Pearson Education. EUA, 2004.

DONG, G., GUO, T. e ZHANG, P. *Security Assurance with Program Path Analysis and Metamorphic Testing*. 4th IEEE International Conference on Software Engineering and Service Science (ICSESS), 2013:193-197.

ERICSON, C. A. *Hazard analysis techniques for system safety*. Wiley-Interscience. New York, 2005:528 p.

ERICSSON. *ERICSSON Mobility Report - On The Pulse Of The Networked Society.* Disponível em http://www.ericsson.com/res/docs/2013/ericsson-mobility-report-june-2013.pdf. Acesso em 23 de Maio de 2015.

F-SECURE Corp. *F-Secure Threat Report H1-2014*. Disponível em https://www.f-secure.com/documents/996508/1030743/Threat_Report_H1_2014.pdf. Acesso em 23 de Maio de 2015.

GARTNER Group. *Worldwide Smartphones Sales to End Users By OS*. Dezembro, 2014. Disponível em http://techcrunch.com/2014/12/15/gartner-301m-smartphones-sold-in-q3-as-xiaomi-muscles-into-the-top-5-at-samsungs-expense/. Acesso em 21 de Maio de 2015.

GELPERIN, D., e HETZEL, B. *The growth of software testing.* Communications of the ACM, Volume 31, Issue 6, June, 1988:687-695.

GOERTZEL, K. M., WINOGRAD, T., HAMILTON, B. A.. *Safety and Security Considerations for Component-Based Engineering of Software-Intensive Systems.* 01 de Fevereiro de 2011. Disponível em https://buildsecurityin.us-cert.gov/sites/default/files/NOSSA-SafeSecureSWComposition-02012011.pdf. Acesso em 20 de Maio de 2014.

GOLD, S. *Android, a secure future at least?* Engineering & Technology Magazine, Apr 09, 2012:50-54.

HARTONG, M., GOEL, R. e WIJESEKERA, D. *Meta Models for Misuse Cases*. Proceedings of the 5th. Annual Workshop on Cyber Security and Information Intelligence Research - CSIIRW '09. ACM, New York, NY, USA, 2009.

HE, F., ZHANG, H. e MEI, T. *A Test Method of Trusted Computing Supporting Software*. The 9th. International Conference for Young Computer Scientists. IEEE Computer Society, 2008: 2330-2334.

HONG, Y., LIU, X., HUANG, S. e ZHENG, C. *Data Oriented Software Security Testing*. 2012 Second International Conference on Instrumentation & Measurement, Computer, Communication and Control. IEEE Computer Society, 2012:676-679.

HOPE, P., ANTÓN, A. I., McGRAW , G. *Misuse and Abuse Cases: Getting Past the Positive*. IEEE Security & Privacy, May/June, 2004:32-34.

HUI, Z., SONG, H., HU, B. e REN, Z. *A Taxonomy of Software Security Defects for SST*. 978-1-4244-6837-9. IEEE, 2010:99-103.

ITU. *ITU Key 2005-2014 ICT data*. Disponível em http://www.itu.int/en/ITU-D/Statistics/Documents/statistics/2014/ITU_Key_2005-2014_ICT_data.xls. Acesso em 22 de Maio de 2015.

KARPPINEN, K., SAVOLA, R., RAPELI, M. e TIKKALA, E. *Security Objectives within a Security Testing Case Study*. 0-7695-2775-2/07. IEEE Computer Society, 2007.

KUMAR, G. e BHATIA, P. K. "*Comparative Analysis of Software Engineering Models from Traditional to Modern Methodologies*". Fourth International Conference on Advanced Computing & Communication Technologies. IEEE Computer Society, 2014:189-196.

LA POLLA, M., MARTINELLI, F. e SGANDURRA, D. "*A Survey on Security for Mobile Devices*." IEEE Communications Surveys & Tutorials Vol.15, Nº 1, First Quarter of 2013:446-471.

LI, G., LU, K., ZHANG, Y., LU, X., e ZHANG, W. *Decoupling Binary-Level Dynamic Test Generation from Specific Architecture Details*. 4th International Conference on Computer

Sciences and Convergence Information Technology. IEEE Computer Society, 2009:1041-1046.

LIPNER, S. *The trustworthy computing security development lifecycle*. 20th Annual Computer Security Applications Conference. Dez, 2004:2-13.

LÚCIO, L., ZHANG, Q., NGUYEN, P. H., AMRANI, M., KLEIN, J., VANGHELUWE, H. e TRAON, Y. *Advances in Model-Driven Security.*Advances in Computers Science: an International Journal, Aug 10, 2013:1-55.

McDERMOTT, J. e FOX, C. *Using abuse case models for security requirements analysis*. Proceedings of 15th Annual Computer Security Applications Conference (ACSAC '99), 1999:55-64.

McGRAW, G. *Bridging the gap between software development and information security*. IEEE Security & Privacy, September/October of 2005:75-79.

MUNASSAR, N. e GOVARDHAN, A. *A Comparison Between Five Models Of Software Engineering*. International Journal of Computer Science Issues. Vol. 7, Issue 5, September 2010:94-101.

MURTHY, K. K., KALPESH, R., THAKKAR, K. R., e LAXMINARAYAN, S. *Leveraging Risk Based Testing in Enterprise Systems Security Validation*. 1st International Conference on Emerging Network Intelligence. IEEE Computer Society, 2009:111-116.

MYERS, G. *The Art of Software Testing*. Hoboken, New Jersey: Word Association, Inc., 1979.

MYLONAS, A., DRITSAS, S., TSOUMAS, B.e GRITZALIS, D. (2011). *Smartphone security evaluation: The malware attack case*. Proceedings of the International Conference on Security and Cryptography (SECRYPT), Seville, Spain, July 2011: 25-36.

NASA. *ISS Risk Summary Card - ISS Program Risk Management*. Symposium on Risk, Hampton, Virginia, May 9, 2001. Disponível em http://www.hq.nasa.gov/office/codeq/risk/docs/incose.pdf. Acesso em 12 de Junho de 2015.

NAUR, P. e RANDELL, B. *Software Engineering: a Report on a Conference Sponsored by NATO Science Committee*. NATO, 1969.

PIEC, M., POZNIAK-KOSZALKA, I., KOZIOL, M. *Analysis of Selected Aspects of "IBM I" Security*. International Joint Conference CISIS'12-ICEUTE´12-SOCO´12 Special Sessions. Advances in Intelligent Systems and Computing. Volume 189, 2013:207-214.

PRESSMAN, R. *Engenharia de Software*. São Paulo: Makron Books, 2005: 1056 p.

RASPOTNIG, C., e OPDAHL, A. *Comparing risk identification techniques for safety and security requirements*. The Journal of Systems and Software nº 86. 2013:1124-1151.

RIBEIRO, R. A. *Segurança no Desenvolvimento de Software*. Rio de Janeiro: Campus, 2002: 310 p.

RIUNGU, L. M. , TAIPALE, O. e SMOLANDER, K. *Software Testing as an Online Service: Observations from Practice*. 3rd International Conference on Software Testing, Verification, and Validation Workshops. IEEE Computer Society, 2010:419-423.

SALINI, P. e KANMANI, S. *Survey and analysis on Security Requirement Engineering*. Computers and Electrical Engineering nº 38. 2012:1785-1797.

SCHWABER, K., SUTHERLAND, J. *SCRUM guide, a definitive guide to SCRUM: rules of game*. Scrum.org, 2013.

SDLC TUTORIAL. *Software Development Life Cycle.* Tutorials Point. Disponível em http://www.tutorialspoint.com//sdlc/index.htm. Acesso em 05 de Maio de 2015.

SETHI, G., DHARANI, A., e PATIL, A. V. *A Survey on Component-Based Software Development System*. Journal of Emerging Technologies and Innovative Research, Vol. 1, Nº 5, October 2014:317-319.

SHABTAI, A., FLEDEL, Y., KANONOV, U., ELOVIC, Y., e DOLEV, S. *Google Android: A Comprehensive Security Assessment*. IEEE Computer and Reliability Societies, Mar/Apr 2010:35-44.

SINDRE, G. e OPDAHL, A. L. *Eliciting Security Requirements by Misuse Cases*. Proceedings of 37th Intl Conf. Technology of Object-Oriented Language and Systems (TOOLS PACIFIC 2000). IEEE Press, 2000:120-131.

SOUISSI, S., SERHROUCHNI, A. *AIDD: A novel generic attack modeling approach*. High Performance Computing & Simulation (HPCS), July 21-25. IEEE Conference Publications, 2014:580-583.

SZONGOTT, C., HENNE, B., SMITH, M. *Evaluating the threat of epidemic mobile malware*. IEEE 8th. International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob). IEEE, Barcelona, Spain. Oct, 2012:443-450.

TAURION, C. *Sua empresa está preparada para o BYOD?* IBM developerWorks. Disponível em https://www.ibm.com/developerworks/community/blogs/ctaurion. Acesso em 23 de Maio de 2015.

TRACY, K.W. *Mobile Application Development Experiences on Apple's iOS and Android OS*. IEEE Potentials, Vol 31, Issue 4, Jul-Aug of 2012:31-34.

TÜRPE, S. *Security Testing: Turning Practice into Theory.* IEEE International Conference on Software Testing Verification and Validation Workshop (ICSTW'08). IEEE Computer Society, 2008.

TSUI, F. e KARAM, O. *Essentials of Software Engineering*. 2nd Edition, Jones and Bartlett Publishers, Sudbury, Massachusetts, EUA, 2011.

TZU, S e PIN, S. *A arte da Guerra - Edição Completa*. 3ª Edição, Saraiva, São Paulo, Brasil, 2014.

VERDON, D. e McGRAW , G. *Risk Analysis in Software Design.* IEEE Security & Privacy, May/June of 2004:32-37.

WANG, X., SOLÍS, C. *A Study of the Characteristics of Behavior Driven Development.* 37th. EUROMICRO Conference on Software Engineering and Advanced Applications - SEAA, 2011: 383-387

WHITTLE, J., WIJESEKERA, D., e HARTONG, M. *Executable Misuse Case for Modeling Security Concerns*. 30th. International Conference on Software Engineering - ICSE '08. ACM/IEEE, 2008: 121-130.

XU, D., TU, M., SANFORD, M., THOMAS, L., WOODRASKA, D. e XU, W. *Automated Security Test Generation with Formal Threat Models*. IEEE Transactions on Dependable and Secure Computing, Vol. 9, Nº 4, July/August, 2012a: 526-540.

XU, W., DENG, L. e ZHENG, Q. *Annotating Resources in Sequence Diagrams for Testing Web Security*. 9th International Conference on Information Technology - New Generations. IEEE Computer Society, 2012b:874-875.

ZHOU, Y., e JIANG, X. "*Dissecting Android Malware: Characterization and Evolution*". Department of Computer Science. North Carolina State University. Disponível em http://www.csc.ncsu.edu/faculty/jiang/pubs/OAKLAND12.pdf. Acesso em 07 de Fevereiro de 2013.

Apêndices

Apêndice A – Anatomia de *Malwares.*

Neste apêndice é apresentado um estudo de alguns *Malwares* realizado de acordo com o modelo proposto – o *SDD* – com o intuito de prover conteúdo para uma base de conhecimento acerca destas ameaças. São abordados alguns tipos de *malwares* de acordo com a classificação da qual trata o Capítulo 2.

O vírus **CABIR** foi um dos primeiros casos de *malware* para dispositivos móveis, identificado em 2004. Trata-se de um vírus que ativa periodicamente a interface *Bluetooth* do dispositivo e a monopoliza, impedindo o uso para outras finalidades. Assim que é instalado, ele inicia-se automaticamente. O vírus também se inclui na seqüência de inicialização do sistema (via mecanismo de "Reconhecimento MIME ") para que ele seja ativado sempre que o dispositivo móvel for ligado, quando então exibe uma caixa de mensagem. O vírus também é espalhado com diversas versões e nomes, como por exemplo, Caribe.sis, EPOC.Cabir (NAV), EPOC_CABIR (Trend), Symbian.Cabir.gen, Symbian/Cabir.a, Symbian/Cabir.b, Symbian/Cabir.rsc e Worm.Symbian.Cabir (AVP).

O **Pmcryptic** é vírus polimórfico que tenta se espalhar por meio de cartões de memória e realizar ataques de negação de serviço. Descoberto em 2008, esse vírus tenta:

- Executar um ataque de negação de serviço para o número 1860;
- Evitar qualquer entrada do usuário de que está sendo recebido
- Copiar-se para a pasta de 2577 com o nome de arquivo "autorun.exe" em todos os cartões de memória encontrados;
- Copiar-se para \ system.exe e definir o atributo “oculto”;
- Lançar o explorador de arquivos do Windows ;
- Exibir um caixa de mensagem com a mensagem de erro "目录 损坏 (Pasta ou diretório) deixou de funcionar."
- Encontrar todos os diretórios e criar uma versão nova polimorficamente modificada (criptografada) de si mesmo como <dirname>. exe e definir o atributo oculto
- Mudar as configurações de cor do sistema;

O virus **Beselo**, registrado em 2008, é distribuído por meio de arquivos de imagem, como "beauty.jpg", por exemplo. Embora a extensão seja a de um arquivo de imagem, o instalador ainda vai reconhecer o arquivo e tentar instalar. O *malware* também tenta se disfarçar

como outros tipos de arquivos de mídia com nomes de arquivos sugestivos, tais como "love.rm" e "sex.mp3". Ele tenta propagar-se por meio de MMS, enviando uma mensagem MMS para cada número na lista telefônica do dispositivo a cada dois minutos. Se a conexão GPRS está desativada então tentará se espalhar por meio de Bluetooth. O *malware* não mantem qualquer controle de dispositivos já infectados e continuará a enviar-se via Bluetooth para os dispositivos próximos. O *malware* também tenta evitar sua exclusão copiando-se para o cartão de memória. Ele copia um arquivo de MDL para o diretório \ Recogs \ \ System, a fim de executar-se na inicialização. Se qualquer um dos componentes do *malware* (EXE, SIS, MDL) foram apagados, o vírus os restaura.

Lascon é um vírus de 2005 que se espalha por meio de redes *Bluetooth*. Ele assume o controle do canal de comunicação e começa a procurar telefones próximos para contaminá-los.

O **FakeFlash.C**, de 2012, é um Trojan em distribuído em formato de aplicação para Android que tenta cobrar uma taxa para o download e instalação do Adobe Flash Player. Quando é executado, abre uma página do site *PayPal* solicitando o pagamento, embora o Adobre Flash Player esteja disponível para download gratuito no site da Adobe. Tem suas variantes com os nomes de *Android.Trojan.FakeFlash* ou *Android/FakeApp* e ataca o sistema operacional Android. O método de infecção é por meio de links em páginas web, embora já tenha sido encontrado na lista de aplicações da Google – *Google Play*.

O Trojan **Gidix.A**, detectado em Novembro de 2013, simula um aplicativo gerenciador de configurações do sistema Android, porém quando ativado o aplicativo carrega os dados sensíveis do dispositivo para um servidor remoto, e também envia silenciosamente mensagens SMS e monitora as chamadas e mensagens SMS enviadas pelo usuário. É encontrado também com os nomes Gidix, KRSMS-A e MisoSMS-B. É distribuído na forma de um aplicativo gerenciador de configurações do sistema chamado adv Service. Como parte do processo de instalação, o aplicativo solicita acesso de administrador do dispositivo, o qual pode permitir a ele bloquear o dispositivo. Uma vez instalado, o aplicativo envia mensagens SMS sorrateiramente. Para disfarçar essa atividade, ele também monitora as mensagens SMS e as chamadas de telefones e verifica se o prefixo de origem é 82 ou 010, possivelmente indicando que este aplicativo foi direcionado para usuários sul-coreanos. O aplicativo limpa o registro de chamadas e de todas as mensagens correspondente a estes números. Além disso, o aplicativo envia disfarçadamente as seguintes informações do dispositivo para um servidor remoto:

- Registros de telefonemas;

- Mensagens SMS;
- Número do telefone;
- Nomes de redes.

Para se proteger o trojan usa o ofuscamento da APKProtection, e também contém o código relacionado com a criptografia e comunicação de uma biblioteca nativa do Android.

O **AVPass.C** , de Janeiro de 2014, é um Trojan distribuído sob a forma de uma aplicação de nome “Clock” para o sistema operacional Android. Enquanto ativo, no entanto, ele rouba informações do dispositivo e tenta desinstalar ou desativar os aplicativos relacionados à segurança instalados no dispositivo. Os ícones são chamados de "atualização do sistema", "360Antivirus" e "QQ"; os últimos dois nomes referem-se a aplicativos populares, mas nenhum dos ícones realmente levam a um programa. Ao clicar em qualquer um dos ícones inicia-se o trojan disfarçadamente em segundo plano, enquanto que os ícones de aplicativos recém-criados são excluídos. Enquanto isso o trojan coleta os seguintes detalhes do dispositivo:

- Mensagem SMS e registo de chamadas;
- Dados de localização GPS;
- Dados da lista de contatos;
- Fotos armazenadas;
- Registro de dados do telefone;

Além disso, o trojan verifica os aplicativos de segurança / antivírus que constam em uma lista interna sua para ver se estão instalados no dispositivo. Se encontrado, o trojan requisita os privilégios de *root user* para desinstalar o aplicativo de segurança / antivírus ou fazê-lo ignorar a si, modificando o banco de dados daquele aplicativo para evitar a sua detecção.

O trojan **Fakeinst.HB** é um clone reembalado de um popular jogo de corrida de carros de acesso gratuito. Ao contrário do original, o clone reembalado exige que o usuário pague uma taxa, supostamente para “acessar níveis mais elevados” do jogo. O jogo original é um jogo de corridas de carros altamente popular para o Android, disponível completamente livre na Google Play Store. O trojan, porém, inclui uma rotina que exige pagamento para continuar a jogar. Durante o jogo o trojan exibe uma mensagem de alerta a cada dez minutos, informando ao usuário que ele deve pagar, supostamente para obter acesso aos níveis mais elevados do jogo. A escala de valores de pagamentos é variável e é apresentada na moeda vietnamita Dong. O usuário é instruído a fazer o pagamento seja por cartões de pré-pagos ou por SMS. Se o usuário

optar por enviar um SMS, o aplicativo irá enviar uma mensagem SMS para um determinado número. Se o usuário se recusa a pagar e ignora as mensagens repetidas, após um período de tempo é direcionado para o site do reempacotador, no qual são mostrados mais jogos (originalmente livres).

O adware **Counterclank.A** é um componente de publicidade usado em vários aplicativos suportados por anúncios. Durante a execução, o componente discretamente recolhe os dados do dispositivo e encaminha-os para um local remoto. O módulo de publicidade é incluído em vários aplicativos, jogos tipicamente populares ou programas relacionados com tratamento de imagens. Os próprios aplicativos não são maliciosos e são softwares livres patrocinados por anúncios, que simplesmente incluíram o módulo de publicidade para fornecer a receita para os desenvolvedores de aplicativos. Além de exibir os anúncios, no entanto, o módulo também provoca vazamentos de informações do dispositivo para um local remoto, sem o conhecimento do usuário. Enquanto o aplicativo está sendo executado, o módulo recupera as seguintes informações do dispositivo e encaminha os dados para http:// [...] apperhand.com [...]:

- Fabricante do dispositivo;
- Modelo do dispositivo;
- Versão do dispositivo;
- Número IMEI (International Mobile Equipment Identity);
- Versão do sistema operacional;
- Configurações locais do telefone;
- Número do telefone;
- Endereço IP de origem;
- User Agent.

O módulo também adiciona um ícone de pesquisa para a tela inicial do dispositivo e um atalho no navegador padrão do dispositivo. O ícone e o atalho têm fins publicitários, e levam o usuário a acessar um suposto provedor de busca de dispositivos móveis em http:// [...] searchmobileonline.com [...].

O risktool **SmsReg.A**, de março de 2013, é comercializado sob o nome de “Battery Improve”, e seu apelo comercial é ajudar a maximizar o uso da bateria do dispositivo. Sem o conhecimento do usuário, o aplicativo também recolhe as seguintes informações:

- Chave da API
- ID da aplicação

- Portador
- Fabricante do dispositivo
- Modelo do dispositivo
- Localização GPS
- Número de Identidade Internacional de Equipamento Móvel (IMEI)
- Operador de rede
- Nome de software
- Versão do SDK

O **BaseBridge.A** é um vírus identificado em Junho de 2011 que atinge o sistema operacional Android. Ele encerra os processos de determinados aplicativos de segurança e tenta enviar SMS e também acessar a internet. Requer a instalação intencional do usuário e é distribuído como um arquivo APK chamado "anserverb_qqgame.apk".

O **Loozfon.A** é um vírus que ataca o sistema operacional Android e envia informações sensíveis para um servidor remoto. Descoberto em Agosto de 2012, esse *malware* requer a instação intencional pelo usuário do dispositivo, geralmente por meio de um link. Android / Loozfon.A é distribuído em sites japoneses voltados para usuárias. O vírus solicita as seguintes permissões: CALL_PHONE, INTERNET, READ_PHONE_STATE, READ_CONTACTS, ACCESS_NETWORK_STATE, e cria um ícone no menu principal do dispositivo. Uma vez que é executado o Loozfon.A mostra um texto em japonês e em paralelo posta informações confidenciais (lista de contatos, e-mail, número de telefone) em uma URL.

Apêndice B – O processo de desenvolvimento de software

O processo de produção e de manutenção de software é composto de um conjunto padronizado de atividades e organizado em sequências e fases distintas, adaptadas aos diversos modelos de projeto e de gestão. O objetivo final é a produção de software de qualidade, com uma razoabilidade de custo e dentro de um prazo específico, de forma que atenda às necessidades e expectativas dos usuários.

Este Apêndice aborda os conceitos da Engenharia de Software aplicáveis ao SDLC – *Software Development Lifecycle* ou Ciclo de Vida do Desenvolvimento de Software, nos diversos modelos presentes na literatura e utilizados pela indústria do software e pelas organizações em geral.

B.1 Modelos clássicos ou tradicionais

Embora as atividades do processo de desenvolvimento de software tenham se mantido praticamente imutáveis desde o início, as técnicas e os métodos empregados para levar a termo um projeto de software têm evoluído e se multiplicado, trazendo inúmeros benefícios. Porém uma dificuldade foi adicionada ao processo, quer seja, a seleção de uma delas para a aplicação no projeto. Este tópico não tem o intuito de analisar ou comparar modelos de desenvolvimento, mas sim de prover uma breve apresentação das suas características principais.

Os modelos ou métodos tradicionais ou clássicos apresentados a seguir refletem as descrições e a avaliação realizada nos trabalhos de Munassar e Govardhan (2010), comparando as características dos cinco principais modelos de desenvolvimento, nas definições e análise de Tsui e Karam (2011), e também no estudo de Kumar e Bathia (2014), utilizados como referência para as considerações apresentadas. O detalhamento das fases de cada modelo são resultado do estudo apresentado no SDLC Tutorial (2015).

B.1.1 Modelo em cascata

O modelo em cascata é formado por um fluxo linear e sequencial de fases. Por isso também pode ser chamado de linear-sequencial, no qual cada fase somente inicia quando a fase anterior estiver completa, de modo que a entrada ou insumo de cada fase é o produto ou entrega da fase imediatamente anterior. As fases são apresentadas na Fig. 15 e descritas a seguir:

- Análise de Requisitos: Nesta fase todos os requisitos do software a ser desenvolvido são identificados e documentados em um documento de especificação de requisitos;

- Projeto do Sistema: As especificações de requisitos da fase anterior são avaliadas nesta fase para que seja preparado o projeto do software. O projeto visa especificar necessidades do hardware e do sistema e também definir a arquitetura do sistema.
- Implementação: Nesta fase as informações de análise e projeto são utilizadas para a codificação dos programas - unidades de software ou *units* - que serão integradas na próxima fase. Cada unidade é desenvolvida e suas funcionalidades testadas em uma atividade denominada teste unitário ou *unit testing*.
- Teste: Nesta fase as unidades de software já testadas individualmente são integradas e o conjunto total do sistema é testado para identificar faltas e falhas;
- Implantação: Uma vez executados os testes funcionais e não funcionais, o produto é implantado no ambiente do cliente ou lançado no mercado;
- Manutenção: A fase de manutenção tem a finalidade de realizar e implantar estas correções ou mudanças (atualizações ou melhorias) no ambiente do cliente.

O modelo em cascata é considerado apropriado para sistemas com as seguintes características:

- Requisitos claros, correto e bem documentados;
- Definição estável do produto;
- Tecnologias conhecidas e estáticas;
- Requisitos sem ambiguidade;
- Recursos com habilidades específicas disponíveis para suportar o produto;
- Projetos curtos.

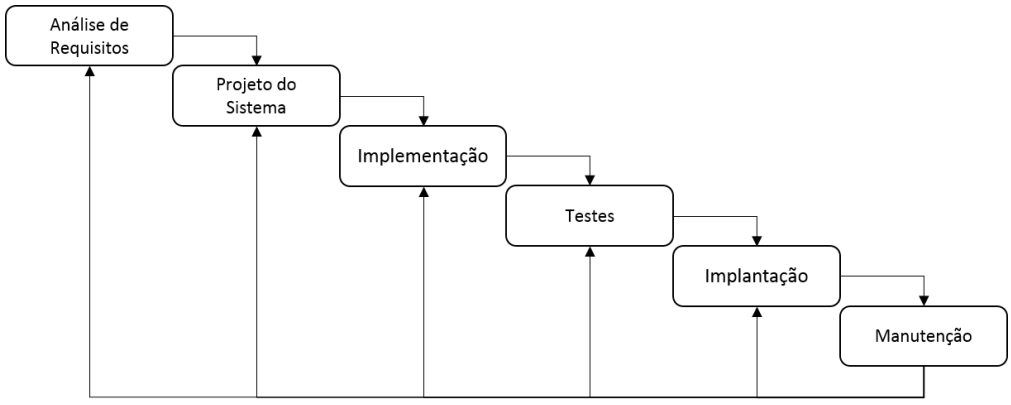


Figura 15 - Modelo em Cascata.
Adaptado de Kumar e Bathia (2014).

B.1.2 Modelo "V"

O modelo em V é uma extensão do modelo em cascata, com o acréscimo de uma fase de testes à cada estágio do desenvolvimento. Isto significa que para cada fase do desenvolvimento há uma fase de teste diretamente associada. É um modelo altamente disciplinado, e uma fase não é iniciada sem a conclusão da fase que a antecede.

No modelo V cada fase de teste, planejada em paralelo, correspondente a uma fase de desenvolvimento. Há fases de verificação em um lado do "V" e fases de validação no outro lado. A fase de codificação une os dois lados do "V". As fases de verificação são as seguintes:

- Análise de Requisitos: é a primeira fase do ciclo de desenvolvimento, na qual os requisitos são entendidos do ponto de vista do cliente. Isto ocorre por meio de uma forte interação, com o registro exato e detalhado dos requisitos e das expectativas do cliente, exigindo o cuidado necessário para identificar indecisões ou falta de clareza do cliente. O plano do projeto dos testes de aceitação é feito nesta etapa, com base nos requisitos do negócio, e é usado como insumo para a elaboração dos testes de aceitação.
- Projeto do sistema: com base nos requisitos detalhados e claros, é o momento de projetar o sistema por completo. Isto requer o entendimento e detalhamento completo da configuração de hardware e comunicação necessários para o sistema em desenvolvimento. Ao fazer isto com antecedência deixa-se mais tempo para a execução dos testes posteriormente.
- Projeto de arquitetura: as especificações da arquitetura do sistema são entendidas e projetadas nesta fase. Geralmente mais de uma proposta de arquitetura é elaborada, e a decisão é tomada com base na viabilidade técnica e financeira destas propostas. O sistema é dividido em módulos conforme suas funcionalidades. Esta fase também é conhecida como Projeto de Alto Nível. O intercâmbio de dados e a comunicação entre os módulos e com outros sistemas é claramente entendida e definida nesta fase, e com esta informação projeta-se e documenta-se os testes de integração.
- Projeto de módulos: Nesta fase, também chamada de Projeto de Baixo Nível, o projeto dos detalhes internos de todos os módulos do sistema é especificado. É importante que o projeto mantenha a compatibilidade com os demais módulos da arquitetura do sistema, e com os demais sistemas. Os testes unitários são uma parte

essencial de qualquer processo de desenvolvimento para auxiliar a eliminar as falhas e erros o mais cedo possível, e são projetados nesta fase com base no projeto dos módulos.

Na fase de Codificação os módulos do sistema, projetados nas fases anteriores, são construídos na linguagem de programação mais adequada, com base nos requisitos do projeto e na arquitetura do sistema. A codificação é realizada com base em guias e padrões e revisada diversas vezes para garantir o melhor desempenho antes de gerar a versão final, que é identificada no repositório de código.

Em seguida vêm as fases de Validação do modelo "V", que são:

- Os Testes Unitários, que foram projetados durante a fase de projeto de módulos, são executados com o código nesta fase. Testes unitários são testes em nível de código para auxiliar na identificação e eliminação de problemas em um estágio inicial do desenvolvimento. Porém nem todos os defeitos podem ser identificados por meio destes testes.
- Os Testes de Integração, que estão associados à fase de projeto da arquitetura, e são executados para avaliar a coexistência e a comunicação dos módulos internos do sistema.
- Os Testes de Sistema diretamente associados como a fase de projeto do sistema, validam as funcionalidades por completo, e a comunicação entre o sistema em desenvolvimento com os outros sistemas. A maioria das ocorrências de incompatibilidade com o software e o hardware utilizados são descobertas nesta fase.
- O Teste de Aceitação está associado à fase de análise de requisitos e compreende o teste do sistema no ambiente do cliente. Estes testes revelam incompatibilidades com outros sistemas ou ocorrências relativas aos requisitos não-funcionais, como carga e desempenho.

A sequência de fases do modelo "V" é apresentada na Fig. 16. Este modelo aplica-se ao desenvolvimento dos mesmos tipos de sistemas para os quais são indicados o modelo em cascata, com as seguintes características:

- Requisitos claros, correto e bem documentados;
- Definição estável do produto;

- Tecnologias conhecidas e estáticas;
- Requisitos sem ambiguidade;
- Projetos curtos.

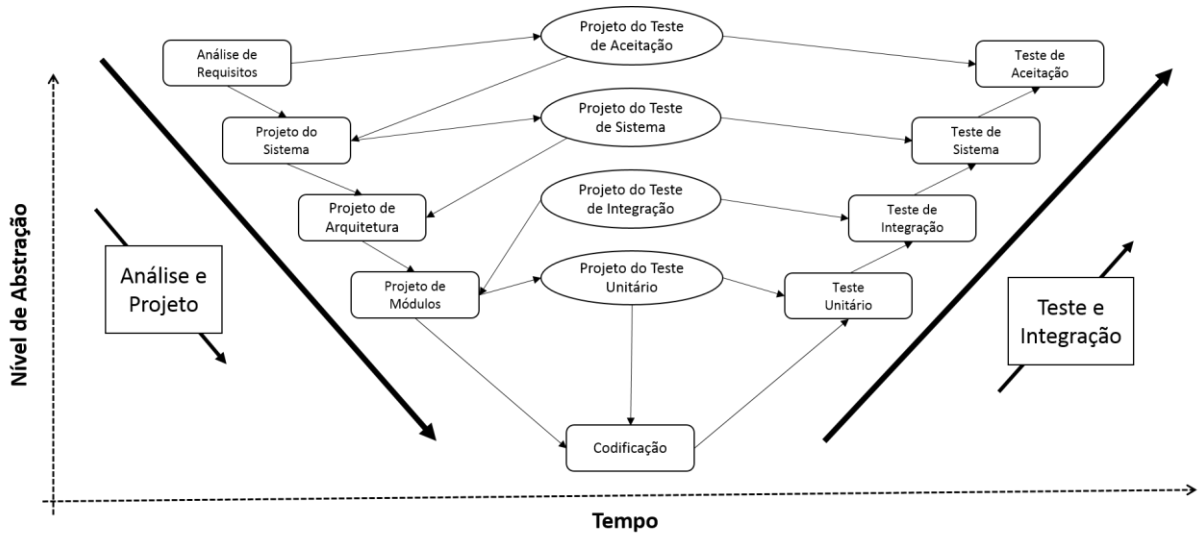


Figura 16 - V-Model.
Adaptado de Munassar e Govardhan (2010).

B.1.3 Modelo Iterativo

No modelo Iterativo não existe a necessidade da especificação funcional estar completa para iniciar-se o desenvolvimento. O desenvolvimento inicia-se com a implementação de um subconjunto de requisitos e iterativamente vai acrescentando funcionalidades e capacidades, até que todo o sistema esteja implementado. À cada iteração são revisados os requisitos e o projeto, sendo então desenvolvidas, testadas e implantadas as novas funcionalidades e capacidades. A ideia básica deste modelo, representado na Fig. 17, é desenvolver todo o sistema por meio de ciclos repetitivos – as iterações – e em pequenas partes de cada vez – os incrementos. Cada incremento é rotulado como uma nova versão ou *Buid.*

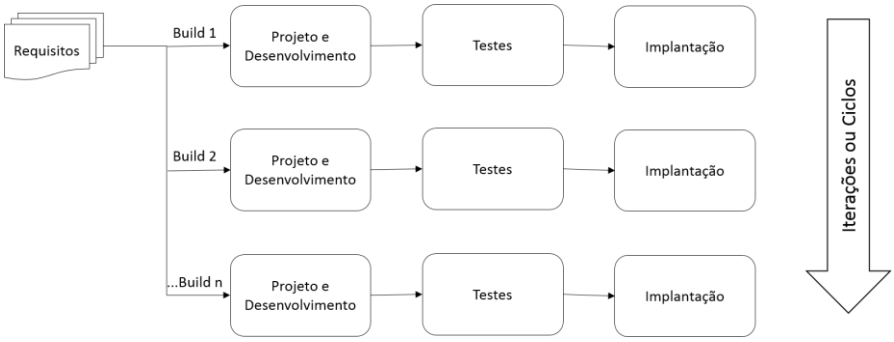


Figura 17 - Modelo iterativo.
Adaptado de Pressman (2005).

Durante o desenvolvimento, mais de uma iteração pode ser conduzida ao mesmo tempo, com a aquisição de informações dos requisitos evoluindo e havendo o incremento de funcionalidades e capacidades à cada versão. O fator chave de sucesso deste modelo é uma rigorosa validação dos requisitos e a verificação e teste da implementação destes requisitos à cada iteração. O modelo é usado normalmente nos cenários com as seguintes características:

- Os requisitos do sistema estão claramente definidos e entendidos;
- A maioria dos requisitos está definida, porém algumas necessidades ou funcionalidades avançadas ainda requerem tempo para ficarem completas;
- Há uma pressão para a conclusão (*Time To Market*);
- Uma nova tecnologia começa a ser utilizada e é necessário o aprendizado por parte do time de desenvolvimento;
- Recursos com conhecimento específico não estão disponíveis, sendo previsto o seu uso para iterações específicas;
- Há funcionalidades que representam alto risco ou objetivos que podem sofrer alterações futuras.

B.1.4 Modelo RAD

O *Rapid Application Development* – RAD – é um modelo de desenvolvimento de software criado pela IBM na década de 70 que utiliza o mínimo de planejamento e uma prototipação rápida. Um protótipo é um modelo que é funcionalmente equivalente a um componente do produto. No RAD os módulos funcionais são desenvolvidos em paralelo com os protótipos e integrados para fornecer o produto completo rapidamente.

O RAD tem como características a obtenção dos requisitos por meio de *workshops* e grupos de foco, o teste precoce de protótipos pelo cliente usando o conceito iterativo, o reuso de protótipos e componentes, a integração contínua e entregas rápidas. O modelo distribui as fases de análise, projeto, construção e testes em uma série de ciclos de desenvolvimentos curtos e iterativos, como mostrado na Fig. 18. As fases no RAD são:

- Modelagem de negócio: O modelo de negócio para o produto é projetado de acordo com o fluxo e a distribuição da informação pelas áreas. A análise de negócio busca identificar informações vitais para o negócio, como é obtida, como e quando é processada e quais fatores impulsionam o fluxo adequado da informação.
- Modelagem de dados: A informação obtida na fase anterior é revista e analisada para formar conjuntos de objetos de dados vitais para o negócio. Os detalhes destas informações e suas relações são identificadas e definidas, bem com sua relevância para o modelo do negócio.
- Modelagem de processos: Os conjuntos de objetos de dados definidos na modelagem de dados são moldados para refletir o fluxo de informação necessário para atingir os objetivos do negócio de acordo com o modelo de negócio. São tratados os processos para qualquer modificação dos dados – inclusão, exclusão, pesquisa ou alteração – de um conjunto de objeto de dados
- Geração da aplicação: O sistema é construído e codificado usando ferramentas de automação para converter os modelos de processo e de dados em protótipos.
- Teste e entrega: O teste geral em RAD é simplificado, uma vez que os protótipos são testados a cada iteração. Entretanto é necessário validar o fluxo de informação e a interface entre todos os componentes. Como esta já foram previamente testados reduz-se o risco de maiores problemas.

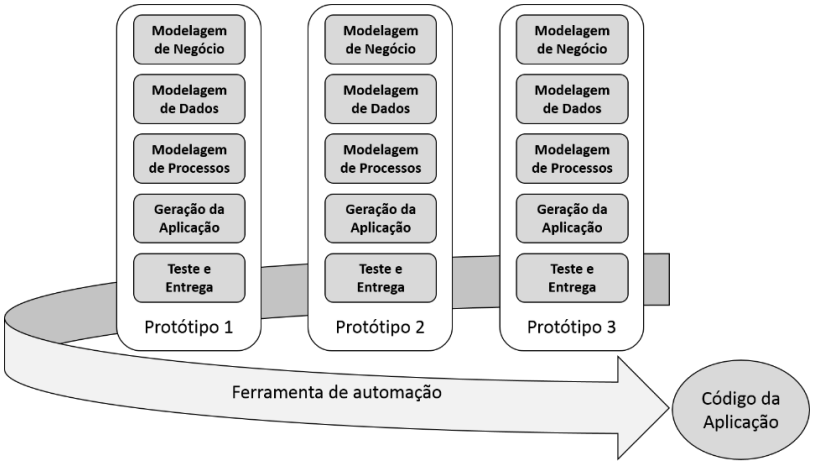


Figura 18 – Desenvolvimento RAD.
Adaptado de SDLC Tutorial (2015).

B.1.5 Modelo do Processo Unificado

Criado pelo trio Jacobson, Booch e Rumbaugh, e mantido e atualizado pela Rational Software Corporation (atualmente IBM) como RUP – *Rational Unified Process* – o modelo Processo Unificado – UP – *Unified Process* é uma abordagem iterativa para o desenvolvimento orientado a objeto. O modelo contempla uma sequência de atividades (workflow) combinada nas seguintes fases:

- Início: Nesta fase é definido o escopo do projeto com base nas necessidades dos interessados;
- Elaboração: Fase na qual o planejamento do projeto é elaborado, com a definição de recursos e arquitetura. Ao término da Elaboração é realizada uma análise dos riscos, da estabilidade da visão e da arquitetura do sistema e das necessidades de recursos;
- Construção: Nesta fase os objetivos são transpostos em documentos de projeto e arquitetura, que são utilizados para a implementação do código;
- Transição: É a fase na qual o sistema é concluído e entregue, sendo realizados o treinamento, o suporte e a manutenção do produto.

No decorrer destas fases as atividades de Modelagem de Negócio, Requisitos, Análise e Projeto, Implementação, Gerência de Configuração, Gerenciamento e Ambiente são executadas em paralelo, como apresentado na Fig. 19.

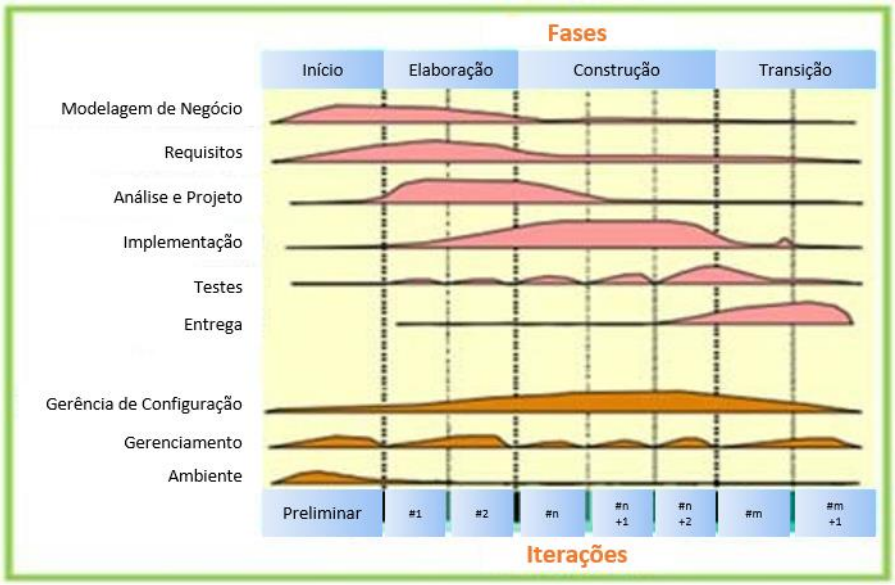


Figura 19 - Modelo RUP.
Adaptado de Kumar e Bhatia (2014).

B.1.6 Modelo em Espiral

Kumar e Bathia (2014) apontam que o modelo em Espiral evoluiu de uma combinação entre os modelos Iterativo e em Cascata, com ênfase na Análise de Riscos. O modelo em espiral, mostrado na Fig. 20, é composto por quatro fases:

- Identificação: É a fase na qual inicia-se a identificação dos requisitos, na base da espiral. À medida em que o produto vai ganhando maturidade os requisitos de sistema, subsistemas e de unidades de software vão sendo preparados nesta fase, nas espirais subsequentes. Isto implica em uma comunicação constante entre os analistas e o cliente. A fase termina com a entrega do software;
- Projeto: Esta fase começa produzindo o projeto conceitual e contempla o projeto de arquitetura, o projeto lógico dos módulos, o projeto físico do produto e o projeto final, nas espirais subsequentes.
- Construção: Trata da construção do software, ou seja, a codificação das unidades de software. No início da espiral, quando o produto ainda está sendo concebido e o projeto está começando, é desenvolvida uma POC – *Proof of Concept* - prova de conceito. À medida em que o projeto avança, trazendo maior clareza e detalhes dos requisitos, um modelo funcional vai sendo produzido, versionado e disponibilizado para o cliente avaliar.

– Avaliação e Análise de Riscos: A análise de riscos contempla a identificação, estimativa e o monitoramento da viabilidade técnica e o gerenciamento de riscos, como variações de prazo e aumento de custos. Nesta fase, após testar uma versão, o cliente faz a avaliação do software e a apresenta para a equipe de desenvolvimento.

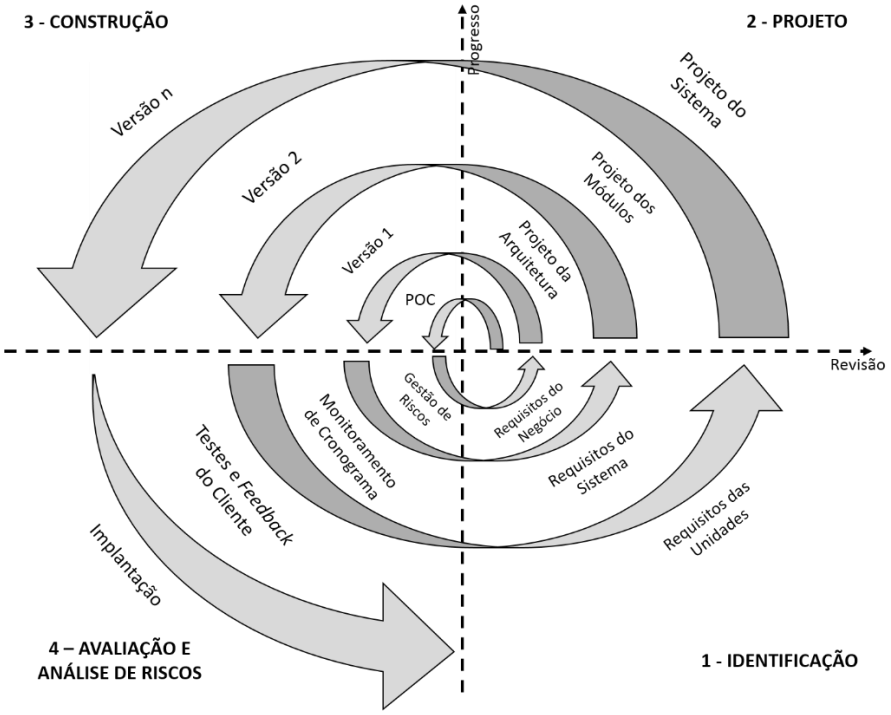


Figura 20 - Modelo em espiral.
Adaptado de SDLC Tutorial (2015).

O modelo em espiral é bastante utilizado na indústria de software por sua compatibilidade com o desenvolvimento natural de qualquer produto, no qual a maturidade advém do aprendizado, o que implica em menor risco tanto para o cliente como para os desenvolvedores. Seus usos típicos envolvem projetos nos quais:

– Existem restrições de orçamento e a avaliação de riscos é importante;
– Os riscos variam de médios a muito elevados;
– O prazo é longo e há riscos de mudanças e prioridades com o passar do tempo;
– O cliente não tem convicção acerca dos requisitos;
– Os requisitos são por demasiado complexos e necessitam de avaliações mais profundas;

- De lançamentos de novos produtos para os quais espera-se o feedback do cliente;
- Há expectativa de alterações significativas do produto durante o ciclo de desenvolvimento.

B.2 Modelos populares e modernos

Os modelos mais recentes e também mais populares são, em geral, modelos desenvolvidos a partir da popularização do paradigma de orientação a objetos e do advento do Manifesto Ágil (Sethi et al, 2014).

Estes modelos têm em comum a mudança de foco, saindo da construção de código e da documentação para o reuso, a integração e as entregas rápidas, além da divisão em camadas, distribuição de funcionalidades e estratificação do software, e a especialização dos desenvolvedores em arquitetura, componentes, interfaces, etc.

B.2.1 Engenharia de Software Baseada em Componentes

O CBSE – *Component Based Sofware Engineering* – Engenharia de Software Baseada em Componentes é um modelo centrado na integração de software, ao invés do desenvolvimento, no qual o software é construído a partir da seleção de blocos denominados componentes. Estes blocos denominados COTS – *Commercial Off-the-Shelf* – ou componentes de prateleira, são selecionados de fornecedores externos ou internos, a partir de repositórios de componentes previamente testados, e montados de acordo com a arquitetura do sistema, reduzindo o custo e o tempo de desenvolvimento. Entretanto isto implica em novas atividades de extrema importância, como a avaliação, a integração e a adequação dos componentes. (Sethi et al, 2014).

Aoyama (1998) estabelece que os componentes do software são unidades binárias cuja produção, aquisição e distribuição é feita de forma independente, e que interagem entre si para formar um sistema funcional, e seu uso no modelo CBSE difere do reuso em outros modelos devido às seguintes características:

- *Plug & play*: Os componentes precisam estar prontos para ser incluídos no software em construção e trabalhar com outros componentes ou *frameworks* sem a necessidade de compilação;
- Centrado em interface: Os componentes devem separar a interface da implementação, de modo que possam ser compostos sem que se conheça detalhes de sua implementação;

- Centrado em arquitetura: Os componentes são projetados para um a arquitetura pré-definida na qual devem operar interagindo com outros componentes e *frameworks*;
- Padronização: A interface dos componentes precisa ser padronizada para que possam ser produzidos por diversos fornecedores e largamente utilizados pelas corporações;
- Distribuição pelo mercado: Os componentes devem ser adquiridos do mercado e melhorados por meio da livre competição e incentivo dos fornecedores.

As fases deste modelo, de acordo com (Sethi et al, 2014) e Ayoama (1998) são as seguintes:

- Análise: Nesta fase os serviços, as restrições e os objetivos do sistema são estabelecidos;
- Seleção de componentes: Paralelamente à análise, os requisitos dos componentes são estabelecidos, e os componentes selecionados de acordo com estes requisitos. O objetivo é prover componentes que possam atender de forma consistente, completa e relevante os requisitos estabelecidos;
- Projeto orientado a componentes: É a fase na qual a arquitetura do sistema é definida e os documentos que definem o comportamento dos objetos preparados, especialmente para objetos com interface múltipla.
- Composição dos componentes: É a construção do sistema em si, com a configuração ou adequação dos componentes de acordo com os requisitos do sistema, a arquitetura e o projeto estabelecidos;
- Teste integrado: É a fase no qual os blocos que compõem funcionalidades completas do sistema, compostos por componentes montados na etapa de composição, são testados para avaliar o atendimento aos requisitos, providenciando os ajustes e correções caso necessário;
- Teste de sistema: É o teste geral do sistema para avaliar se o mesmo atende aos requisitos, e também para identificar e corrigir possíveis falhas na composição ou no comportamento dos componentes.

A Fig. 21 apresenta a sequência de fases do modelo CBSE conforme a descrição apresentada.

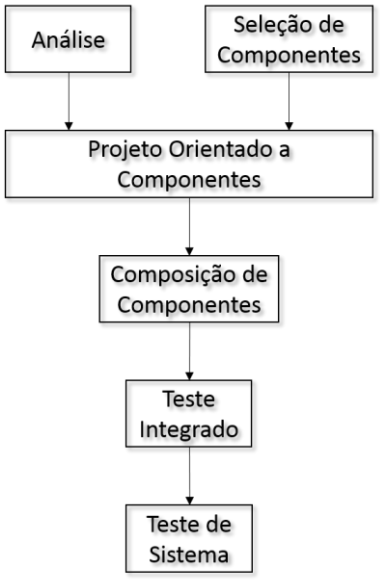


Figura 21 - Fases do CBSE.
Adaptado de Ayoama (1998).

B.2.2 Desenvolvimento ágil

Os métodos de desenvolvimento mais populares e modernos – também denominados simplesmente ágeis ou *Agile* - compõem uma abordagem do SDLC que busca enfrentar a imprevisibilidade por meio de uma estratégia de construção incremental e iterativa do software baseada em *sprints* – conjuntos de pacotes de trabalho que agregam funcionalidades ao software (Kumar e Bathia, 2014). Os métodos são guiados, em geral, pelos princípios do Manifesto Ágil, cujos conceitos chave são:

- A prevalência dos indivíduos e suas interações sobre os processos e as ferramentas;
- Entrega de software executável ao invés de produção de documentação;
- A colaboração do cliente é mais importante do que a negociação de contratos;
- A resposta rápida às mudanças é melhor que seguir planos.

Os métodos ágeis são uma resposta aos modelos clássicos excessivamente burocratizados que dificultam a atuação de profissionais de elevada maturidade e priorizam o controle em detrimento da entrega (Sdlc Tutorial, 2015). A seguir são apresentados alguns dos mais conhecidos e utilizados exemplos destes métodos.

B.2.2.1 Extreme Programming

A Programação Extrema ou XP – *Extreme Programming*, cujas fases são mostradas na Fig. 22, é um modelo de desenvolvimento de software que contempla cinco princípios fundamentais:

1) O feedback rápido, resultado da programação em pares, teste unitário, integração e iterações curtas gerando frequentes versões do software;
2) A simplicidade, minimizando as preocupações com o porvir;
3) A mudança incremental, fazendo pequenas mudanças que acrescentem algo ao software, e também por meio da refatoração do código;
4) O apoio à mudança, deixando opções para decisões futuras e postergando decisões críticas para momentos mais oportunos;
5) A qualidade do trabalho, buscando criar o melhor produto possível – partindo da premissa de que essa é uma tendência natural dos programadores.

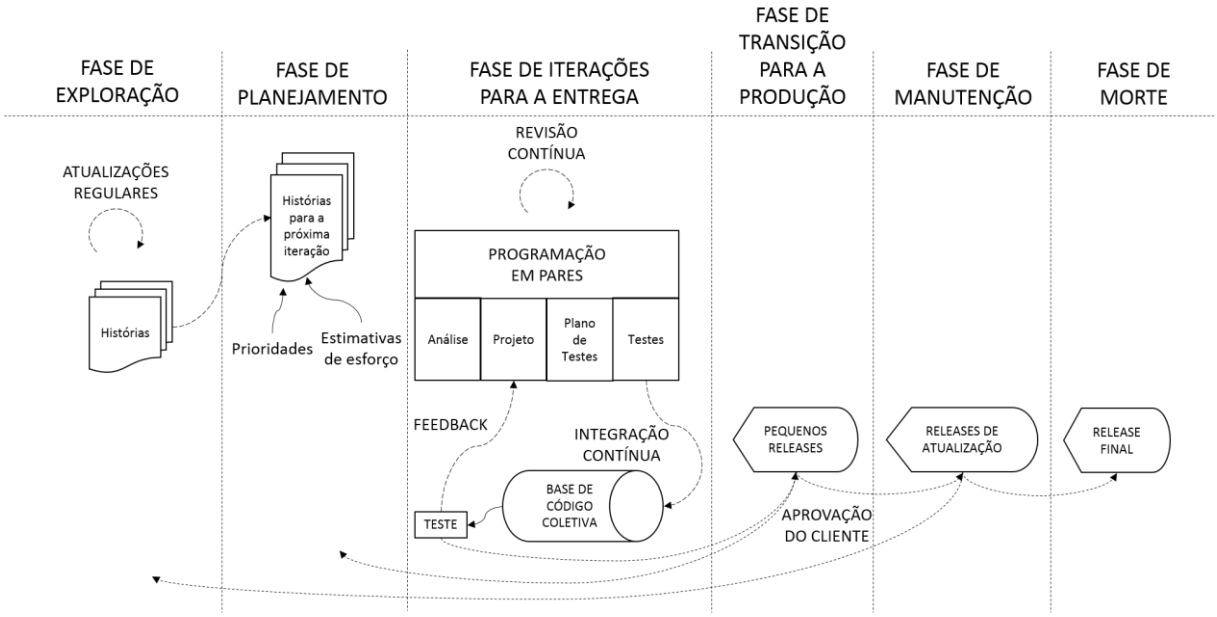


Figura 22 – Extreme Programming (XP).
Adaptado de Kumar e Bathia (2014).

Kumar e Bathia (2014) destacam que, usada por pequenos times, normalmente atuando no mesmo espaço físico para facilitar a comunicação e a cooperação, a XP propõe a criação do mínimo de documentação possível, sendo o código e os testes parte complementar da documentação. Isto reforça a intensa troca de informações entre os membros do time de desenvolvimento e o cliente do software, a simplicidade na documentação e no código, as várias formas de *feedback* e decisões corajosas e rápidas.

Em decorrência dessas características a XP apresenta algumas vantagens, como o fato de possibilitar rapidamente mudanças significativas para a melhoria do produto, contar com alto grau de maturidade e expertise do time de desenvolvimento e simplificar ao máximo a

solução de problemas, o que é positivo do ponto de vista da segurança da informação no processo de desenvolvimento de software (SDLC Tutorial, 2015).

Por outro lado, a ausência de documentação e da análise e gestão de risco, o fraco planejamento e a alta dependência do desempenho e conhecimento individual fazem com que aumente em muito o risco de falhas, notadamente aquelas latentes, que serão manifestadas somente em momentos críticos após o final do desenvolvimento. Há também o risco da instabilidade pela pressão por entregas cada vez mais rápidas, o que reduz o tempo para a validação e verificação (SDLC Tutorial, 2015).

B.2.2.2 SCRUM

O SCRUM, cuja sequência de atividades é mostrada na Fig. 23, é um modelo flexível utilizado por equipes com alto nível de expertise e maturidade com o intuito de prover entregas de *software* rápidas e de qualidade. O produto de software é divido em blocos ou módulos que passarão a compor o *backlog* de produto. Estes módulos são decompostos em funcionalidades completas denominadas *Sprints*, que são avaliadas diariamente e devem ser desenvolvidas completamente em períodos de até trinta dias, disponibilizando funcionalidades completas e operacionais do software (Schwaber e Sutherland, 2013).

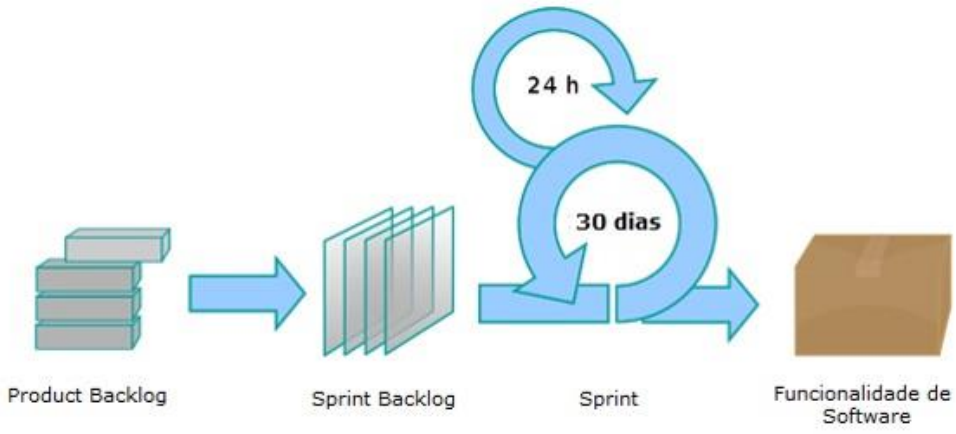


Figura 23- O modelo SCRUM.
Fonte: http://www.devmedia.com.br/introducao-ao-scrum.

De acordo com Schwaber e Sutherland (2013), o SCRUM pode ser utilizado em conjunto com outros modelos e técnicas com o intuito de resolver problemas complexos cuja solução possa ser tratada de forma incremental. Apresenta-se como uma boa alternativa para projetos nos quais o prazo é curto e crítico e os requisitos mudam com frequência.

B.2.2.3 Crystal Clear

De acordo com Cockburn (2004) o modelo Crystal[7] Clear é um componente do conjunto de metodologias Crystal, considerada uma metodologia ágil ou leve, recomendado para equipes de até 6 ou 8 desenvolvedores que trabalham no mesmo espaço e em sistemas que não são críticos - especialmente para a segurança das pessoas. O conjunto de metodologias Crystal tem seu foco na eficiência e habitabilidade como componentes de segurança do projeto. O Crystal Clear se concentra em pessoas, e não processos ou artefatos.

O Crystal Clear demanda as seguintes características, denominadas propriedades:

- Entrega frequente de código utilizável para os usuários;
- Melhoria refletiva;
- Comunicação osmótica de preferência, devido à ocupação do mesmo ambiente;
- Segurança pessoal;
- Foco;
- Fácil acesso a usuários experientes;
- Testes automatizados, gerenciamento de configuração e integração frequentes.

Cockburn (2004) desenvolveu a abordagem do modelo Crystal com o foco nas pessoas, na interação, na comunidade, nas habilidades, nos talentos e nas comunicações com a crença de que estes são o que tem o efeito de primeira ordem sobre o desempenho. Os processos são importantes, porém secundários.

A filosofia de Cockburn (2004) se traduz em um reconhecimento de que cada equipe tem um conjunto diferente de talentos e habilidades e, portanto, cada equipe deve usar um processo exclusivo feito sob medida. Isso significa que o processo deve ser minimizado – sua importância é pouco significativa.

Cockburn (2004) apresenta as diferenças entre a metodologia, as técnicas e as políticas. A metodologia é um conjunto de elementos (práticas, ferramentas); as técnicas são áreas de habilidades, tais como o desenvolvimento de casos de uso; e as políticas definem as obrigações organizacionais.

[7] A utilização do termo *Crystal* - "cristal" refere-se às várias faces de uma pedra preciosa - cada face diferente em um núcleo subjacente. O núcleo subjacente representa os valores e princípios, enquanto cada face representa um conjunto específico de elementos, tais como técnicas, funções, ferramentas e padrões, que são aplicáveis à medida em que a complexidade e os custos do projeto tornam-se mais significativos

B.2.2.4 Feature Driven

O desenvolvimento guiado por funcionalidades ou FDD – *Feature Driven Development* é um modelo de desenvolvimento ágil de software que busca uma entrega constante de resultados efetivos e funcionais durante todo o processo de desenvolvimento, entretanto sem concentrar-se excessivamente nas etapas de codificação e testes (Kumar e Bathia, 2014). O modelo utiliza diversas práticas de XP como os testes unitários, refatoração, programação em pares, integração contínua e outras, mas também define a inspeção formal de projeto e de código e a posse de código ou classe, no que se confronta com o modelo XP.

Conforme mostrado na Fig. 24, as atividades são divididas em cinco processos principais:

1) O desenvolvimento de um modelo abrangente, usando ferramental e técnicas de levantamento de requisitos, análise orientada a objetos e modelagem de dados, com o intuito de produzir um modelo de alto nível contemplando objetos e/ou dados, o qual será utilizado como referência durante todo o processo de desenvolvimento;

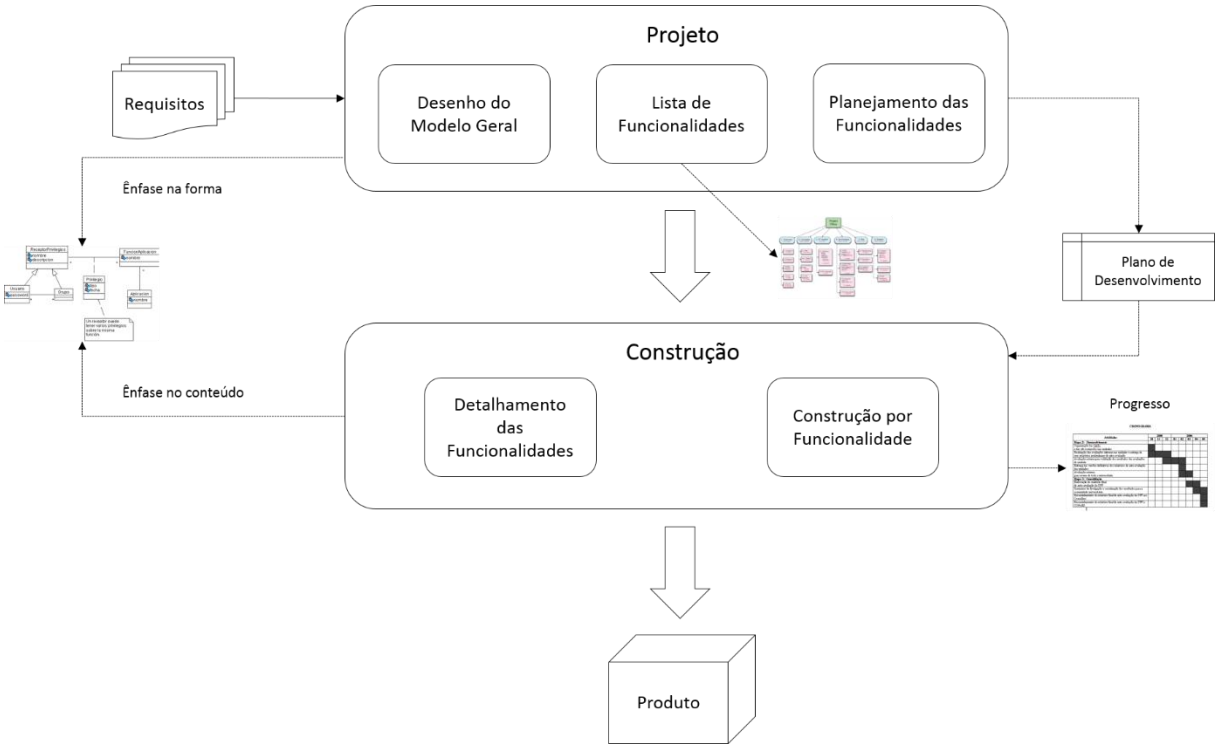


Figura 24- FDD – Feature Driven Development.
Adaptado de Kumar e Bathia (2014).

2) A construção da lista de funcionalidades, dividindo o modelo de alto nível em áreas de negócio, atividades e passos a serem automatizados, uma hierarquia que representará o produto final de software chamada de *product backlog*;
   O planejamento das funcionalidades definindo os pacotes de trabalho na sequência mais adequada para a construção e levando em conta a complexidade e a dependência entre as mesmas;
3) O detalhamento das funcionalidades, gerando os artefatos e *templates* para a codificação de cada funcionalidade, incluindo-se aí os testes a serem empregados;
4) A construção por funcionalidade, na qual cada *template* de código é completado, testado e inspecionado e o resultado é incorporado ao projeto já em condições de uso pelo cliente do produto.

B.2.2.5 Test Driven

O desenvolvimento guiado por testes, TDD - *Test Driven Development*, é uma técnica de desenvolvimento de software do modelo ágil que parte da escrita de casos de teste para a elaboração do código. Um caso de teste é escrito para uma ferramenta que possibilite testes automatizados – como JUnit, por exemplo – e a partir da execução deste caso de testes e da identificação da falha o código é revisto buscando corrigir a falha. Este processo, mostrado na Fig. 25, é repetido até que o teste seja validado, quando então o código é refatorado para atender aos demais padrões de desenvolvimento.

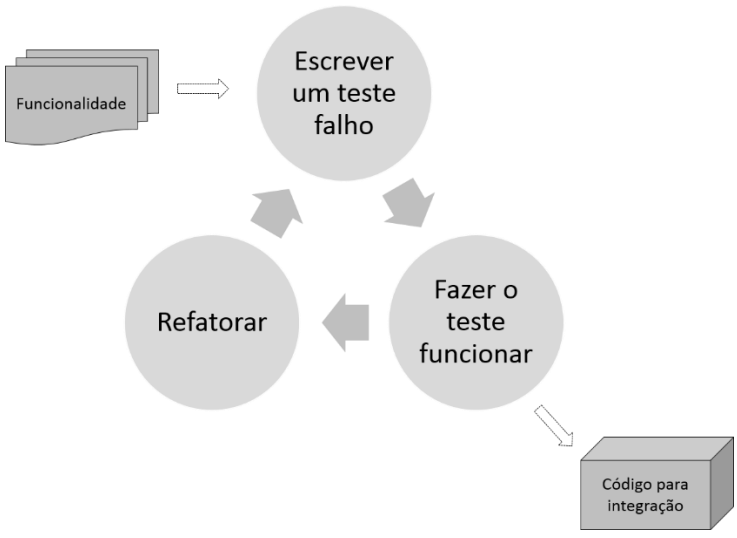


Figura 25 – Test Driven Development.

Do ponto de vista de agilidade e eficiência do processo, e mesmo da qualidade do software no que diz respeito a erros, o TDD é sem dúvida um modelo efetivo e cuja adoção tem ocorrido com entusiasmo pela indústria de software devido às promessas de aumentar a cobertura dos testes e produzir sistemas com baixo acoplamento e alta coesão. Além disso esse modelo força um aprimoramento do escopo – e dos requisitos – e uma constante integração entre as atividades de análise e projeto e as de desenvolvimento.

Em uma análise mais acurada foram identificados alguns aspectos negativos, como a necessidade de maturidade e expertise na elaboração dos casos de teste para não comprometer a cobertura dos testes (Abrahamsson e Siniaalto, 2007). A excessiva ocorrência de ciclos rápidos de mudanças pode gerar um descontrole de custos. Também foi identificada a inadequação para testes de funcionalidades *multithreads* e de segurança do software, pois nesses casos há uma dificuldade natural em comprovar a adequação do código aos requisitos de forma a concluir o ciclo de revisões.

B.2.2.6 Behavior Driven

O desenvolvimento guiado por comportamento – BDD, *Behavior Driven Development* - é uma evolução e uma resposta aos problemas e dificuldades do TDD. O principal objetivo do modelo BDD é obter especificações executáveis do sistema que possam ser aplicadas às ferramentas que suportam o modelo, por meio de uma linguagem ubíqua, de acordo com Wang e Solís (2011). Além desta característica os autores enumeram: o processo de decomposição iterativa dos requisitos de negócio; a descrição dos cenários e estórias do usuário em texto plano; testes de aceitação/homologação automatizados com regras mapeadas; o código legível orientado à especificação do comportamento e o direcionamento por comportamento em fases diferentes.

Porém a constatação feita pelo estudo de Wang e Solís (2011) é que o BDD não é uma unanimidade na indústria de software nem mesmo quanto a suas características. Além da questão da maturidade do modelo, há também um excessivo foco das ferramentas de suporte ao BDD nas etapas de implementação, fazendo com que boa parte do processo de desenvolvimento seja suportado por ferramentas auxiliares ou simples modelos de documentos, especialmente nas etapas de análise e projeto. Essas deficiências são críticas para o desenvolvimento de software seguro, e o estudo propõe um avanço nas ferramentas e no próprio modelo para que venha a atender as expectativas e os propósitos.

B.2.2.7 Model Driven

Decorrente da abordagem de engenharia dirigida por modelo, a MDE - *Model Driven Engineering*, e da arquitetura dirigida por modelo, a MDA – *Model Driven Architecture*, o desenvolvimento dirigido por modelo ou MDD - *Model Driven Development*, cujo processo de transformação é mostrado na Fig. 26, apresenta-se como uma importante alternativa perante os crescentes desafios do desenvolvimento de software.

Uma especialização deste modelo, a segurança dirigida por modelo ou MDS - *Model Driven Security* – vem se tornando uma opção considerável em termos de desenvolvimento para os softwares cujo aspecto de segurança é crítico, pois possibilita a representação da segurança na modelagem do software, a composição desses aspectos em conjunto com o modelo de negócio do software e os testes dos requisitos de segurança já nesses modelos (Nguyen et al, 2014).

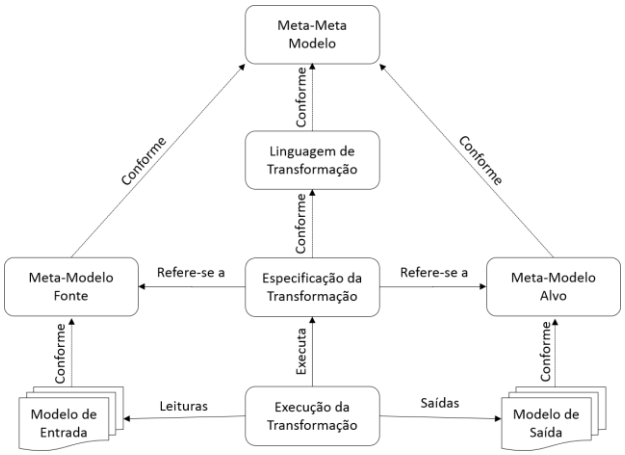


Figura 26 - O processo de transformação do modelo no MDD.
Adaptado de (Lúcio et al, 2013).

O MDE prioriza os modelos do software e suas transformações em cada etapa do processo de desenvolvimento, o que simplifica o grau de abstração necessário em contraposição aos demais artefatos e ao código propriamente dito (Lúcio et al, 2013). A MDE contempla uma abordagem metodológica e um ferramental para elaboração, manipulação e transformação de modelos, os quais, usando a abstração dos processos para os quais o software está sendo desenvolvido, são de mais fácil compreensão para os projetistas e engenheiros do que as linguagens de programação.

Desta forma busca a elaboração de software de melhor qualidade, que tenha maior aderência aos processos de negócio, o que é baseado na premissa de que estes modelos representam de forma mais adequada as necessidades do negócio, enquanto que as linguagens e ferramentas tradicionais estão mais voltadas para a arquitetura e as características dos sistemas computacionais. Deste modo é certamente mais fácil entender, manipular e testar conceitos acerca desses processos nos modelos do que trabalhar com códigos e abstrações de nível mais baixo com o mesmo propósito.

O processo de desenvolvimento concentra-se então nas transformações do meta-modelo em modelos capazes de representar de maneira simplificada o processo de negócio que se pretende automatizar. Disso advém a constatação que o principal trabalho dos engenheiros e dos desenvolvedores de software é prover as transformações de meta-modelos em modelos e o refinamento desses modelos em sucessivas transformações, as quais provêm os demais tipos de artefatos do processo de desenvolvimento de software, tais como diagramas e o próprio código.

A MDD simplifica a compreensão do software e favorece a modelagem dos requisitos de segurança da informação, possibilitando o tratamento precoce desses requisitos no que se refere ao correto entendimento, comportamento frente aos riscos e ameaças já identificados e validação da eficácia das soluções consideradas, assim como os efetivos testes dessas soluções.

O foco desse método é a produção de modelos que representam de forma simplificada os processos a serem automatizados. Esses modelos – que são baseados em meta-modelos - linguagens de descrição de conjuntos de modelos, que também são chamados de formalismos - passam por sucessivas transformações para produzir os diversos artefatos, incluindo diagramas UML e o próprio código fonte.

No que tange a segurança da informação, a MDD avança no sentido da segurança dirigida por modelo – MDS, que vem a ser a modelagem dos requisitos de segurança nos mesmos moldes da modelagem dos processos de negócio. Por tratar-se de método que prioriza a elaboração de modelos e suas transformações em cada estágio do processo de desenvolvimento do software, a MDS reforça o tratamento precoce dos requisitos de segurança da informação. As questões de segurança são endereçadas desde o início do processo, embora sejam tratadas durante todo o ciclo de desenvolvimento.

Além disso os requisitos de segurança tratados no modelo permitem entregar uma implementação segura sob todos os aspectos, independentemente da plataforma e da arquitetura, favorecendo também os aspectos de conectividade e de interoperabilidade do

software. Outro aspecto interessante do ponto de vista da qualidade do software e da segurança da informação é a automação dos processos de transformação dos modelos, que reduz a interferência humana, fator bem conhecido de inserção de erros.

De acordo com a análise realizada por (Lúcio et al, 2013), a MDS apresenta uma boa perspectiva quanto a tornar-se viável para o desenvolvimento de software crítico, no qual o tratamento dos aspectos de segurança da informação é prioridade. A abordagem em perspectiva avalia as técnicas mais utilizadas do método, quer sejam, UMLsec, SecureUML, Sectet, ModelSec e a SecureMDD, considerando as preocupações com a segurança, a abordagem de modelagem, a separação entre requisitos de segurança e do negócio – Separation of Concerns (SoC), as transformações dos modelos, a verificação, a rastreabilidade, as ferramentas de suporte e a validação dos modelos. De acordo com estes autores, cada uma das técnicas apresenta características relevantes sob determinados aspectos, a saber:

- A **UMLsec** enfatiza a preocupação com a segurança nos modelos com o reforço de diversos requisitos de segurança, como confidencialidade, integridade, autenticidade, autorização, atualidade, fluxo seguro de informação, não-repúdio e troca justa. Essa abordagem necessita da modelagem de atacantes – as máquinas adversárias – para simular o comportamento dos mesmos durante a análise do sistema. Como aspectos negativos pode-se ressaltar que a UMLsec não gera código a partir dos modelos e provê rastreabilidade apenas para as sequências de ataques em potencial.
- A **SecureUML** é uma abordagem mais voltada para o controle de acesso com base no RBAC – *Role-Based Access Control.* Além dos tradicionais elementos Sujeito, Regra e Permissão são introduzidas as notações de Recurso e Ação, para os quais são reforçadas as questões relativas à segurança da informação. As propostas e projetos já implementados utilizando Java (EJB) e .NET apresentam resultados expressivos, porém a falta de rastreabilidade e o uso restrito ao controle de acesso são fatores restritivos ao seu uso.
- A **Sectet** é voltada para aplicações distribuídas no conceito SOA implementadas como *web services*, abrangendo as políticas básicas de segurança para as mensagens trocadas pelos componentes distribuídos e também políticas avançadas de controle de acesso estático e dinâmico com RBAC. A separação entre a modelagem de requisitos de segurança e os requisitos de negócio emprega a UML e a OCL – *Object Constraint Language*, e três tipos de visões de *workflow*, sendo um global – para as interações

entre organizações, um local – para os processos de negócio ou componentes de serviço, e um para a interface – para as propriedades e permissões de cada componente de serviço do sistema. A verificação e a rastreabilidade não foram definidas ou implementadas ainda.

- A **ModelSEC** abrange a modelagem de código seguro para controle de acesso e uso de base de dados. As preocupações com a segurança implicam no emprego de componentes específicos como Ativos – objetos lógicos ou físicos expostos a riscos; Ameaças – que podem danificar ou afetar os Ativos; Salvaguardas, medidas de restrição ou mitigação de riscos; Planos de contingência – conjuntos de salvaguardas para a redução dos riscos; Requisitos de segurança, que expressam a segurança que o sistema final deve prover. Uma das deficiências apontadas pelos autores é a falta de detalhes quanto à integração dos requisitos da segurança com os requisitos de negócio. A outra refere-se à ausência de suporte à verificação, rastreabilidade e validação dos modelos e de suas transformações.
- A **SecureMDD** é uma abordagem baseada em UML voltada para o desenvolvimento de aplicações embarcadas baseadas em protocolos criptográficos, visando protege-las de ataques de intrusos, tais como interceptação de mensagens ou quebra de confidencialidade e integridade (Moebius et al, 2009). Uma das deficiências desse modelo é a dependência da plataforma para a qual os diagramas UML são elaborados. Isso também influi na rastreabilidade, uma vez que dispensa a rastreabilidade para trás.

Ainda nesta análise (Lúcio et al, 2013) ponderam que, mesmo considerando as dificuldades ainda enfrentadas pela técnica devido ao seu surgimento recente e à falta de ferramentas adequadas, a abordagem dirigida por modelo tem se mostrado como uma proposta interessante no sentido de aprimorar a qualidade e a segurança do software nas fases iniciais do processo de desenvolvimento, sendo progressivamente adotada em áreas tais como a indústria automobilística e de dispositivos de computação móveis.

A conclusão é que o modelo teórico da MDA/MDE/MDD é bastante adequado ao tratamento precoce da segurança de informação no processo de desenvolvimento de software, porém ainda não há um *framework* ou ferramental que possibilite o uso do método em todas as etapas do processo, e tampouco para softwares das diversas indústrias, e de forma padronizada – o que cria complicações para a difusão do método e capacitação de equipes de desenvolvedores.

B.3 Modelos e a segurança da informação

É certo que significativa parcela dos problemas enfrentados por softwares de qualquer natureza decorre das falhas humanas, quer seja no processo de produção do software, quer seja na sua configuração e adequação ao ambiente, quer seja no seu uso. Os diversos métodos, técnicas, ferramentas e abordagens apresentados tem por objetivo final mitigar os riscos inerentes a esse processo e decorrentes dessa premissa. Sua constante evolução, além de buscar maior eficiência em termos de custo e prazo para a produção de software, também objetiva tornar o processo de desenvolvimento de software mais padronizado e seguro. Desta forma procura-se aumentar a confiabilidade do produto final e reduzir a sua dependência do fator humano no que tange a qualidade e a segurança.

Devido à exposição do software aos problemas das diversas naturezas citadas, é imprescindível que haja, no processo de desenvolvimento de software, um planejamento robusto e abrangente, que considere os riscos na proporção adequada à segurança exigida para o software a ser produzido. Também é necessário que haja uma especial atenção aos procedimentos que garantirão a qualidade do software – e de forma especial aos testes a serem realizados durante todo o processo – bem como à padronização e o uso de métodos, técnicas e ferramentas como nível adequado de maturidade requerido pelo software, e também à automação de tarefas e atividades padronizadas, reduzindo a interferência humana e, por conseguinte a possibilidade de geração de problemas.

Segundo (Goertzel et al, 2011) todo produto de software é um conjunto de componentes e funcionalidades que interagem entre si, com o usuário, com o ambiente computacional no qual opera – hardware, sistema operacional, redes, entre outros – e, portanto, dependente do comportamento e do funcionamento de todo este sistema. Em sua análise sobre os impactos dos componentes do software sobre a segurança das pessoas e da informação, (Goertzel et al, 2011) inferem que qualquer anomalia no funcionamento ou no comportamento de um ou mais desses elementos pode causar problemas para o software e consequentemente falhas de segurança – considerada aqui não somente segurança da informação, mas a segurança no aspecto mais amplo, que pode inclusive colocar em risco a vida e a integridade de seres vivos.

Estendendo esta dependência ao processo de desenvolvimento de software, (Goertzel et al, 2011) consideram que há ainda uma dependência decorrente das atividades realizadas naquele processo, desde os primeiros momentos, pois certamente são tomadas decisões e executadas ações que tornar-se-ão cruciais para o funcionamento adequado do software. Essas

decisões e ações contribuem para a qualidade do software, para a sua sustentabilidade e para a segurança das informações a serem tratadas por este, bem como a reação adequada às falhas e o enfrentamento de ameaças.

Além disso, as próprias atividades executadas no processo de desenvolvimento de software podem inserir problemas nele, muitos dos quais de forma latente, que manifestar-se-ão após a sua disponibilização para uso. Há que se considerar também a possibilidade de medidas e contramedidas que se mostrarão inócuas ou insuficientes para conter falhas e repelir ameaças. Alguns destes aspectos são apontados na sequência.

Cascata

Do ponto de vista da segurança da informação as principais vantagens são o planejar antes de fazer, definindo prazos e resultados esperados – os artefatos entregáveis - para cada etapa, e o uso de documentação formal. Por outro lado, alterações ou mudanças não são bem-vindas - mesmo perante uma expectativa de requisitos acurados que não é realista.

O software só é avaliado de forma efetiva e totalmente na fase de integração e testes, quando é tarde para identificar os problemas – especialmente se houve algum atraso nas etapas iniciais que acaba sendo compensado nesta etapa. Há também a dificuldade de integrar o modelo ao gerenciamento de riscos, o que é especialmente complicado para projetos de maior porte (Munassar e Govardhan, 2010).

Iterativo

Uma das vantagens do modelo iterativo é que este facilita a identificação de problemas com a especificação de requisitos e permite correções e mudanças durante praticamente todo o projeto. Por outro lado, é muito difícil dividir um sistema – especialmente um projeto de pequeno porte - de modo a permitir iterações ou ciclos que entreguem funcionalidades completas. Também pode-se considerar que o modelo iterativo é a base das metodologias ágeis. Sob o aspecto de segurança da informação, o mais crítico é que possíveis revisões, mudanças ou falhas nas integrações entre as entregas das iterações resultem na incorporação de erros ou falhas de segurança no software.

RAD

No que tange à segurança da informação a excessiva simplificação e a pressão por resultados é um fator crítico, podendo impactar em falhas latentes mais graves que repercutirão no software quando já em uso. A necessidade de profissionais altamente especializados pode representar

um problema na medida em que restringe o conhecimento daqueles às áreas específicas, ou impacta fortemente nos custos ao prover especialistas de diversas áreas – incluindo-se a segurança da informação – para compor a equipe do projeto.

Modelo do Processo Unificado

No que compete à segurança da informação o comportamento do modelo *Unified Process* é bastante similar ao RAD, porém incorporando as especificidades da orientação a objetos.

Espiral

É um processo bastante natural, na medida em que há uma evolução na maturidade tanto do cliente do software quanto da equipe de projeto, reduzindo os riscos de insatisfação. Em função disso, o fato de adicionar novos requisitos ou promover mudanças somente a partir do momento em que estes estão em um elevado nível de maturidade, juntamente com o software em desenvolvimento, representa um diferencial positivo.

Entretanto há a necessidade de um envolvimento maior do cliente e a sua contínua evolução, bem como o aprimoramento do relacionamento destes com os desenvolvedores. Estes dois aspectos tornam necessário um forte componente de gestão para evitar que o processo entre em uma espiral infinita, jamais apresentando o software por completo.

A incorporação da análise de risco a cada iteração reforça a qualidade, e o software é entregue de forma recorrente, desde muito cedo. Grandes projetos de missão crítica são bem atendidos por este modelo. Porém a segurança da informação pode ser comprometida com as sucessivas mudanças - a gestão de mudanças torna-se um fator de risco, e o custo aumenta. A análise de riscos requer expertise e especialistas, e é fator determinante do sucesso do projeto (Munassar e Govardhan, 2010).

Apêndice C – Modelos/*Templates* e Exemplos de documentos.

C.1 Caso de uso impróprio

O documento a seguir é um *template* para a elaboração de um caso de uso impróprio, de acodo com as descrições e orientações para a elaboração deste documento, as quais foram apresentadas na Seção 4.4.

Id		**Versão**	
Descrição			
Autor		**Data**	
Projeto			
Atores			
Diagrama:			
Fluxo básico:			
Fluxo alternativo:			
Exceções:			
Pré-condição:			
Pós-condição:			
Pontos de captura:			
Pontos de extensão:			
Requisitos relacionados:			
Características do atacante:			
Interessados e Risco:			

C.1.1 Elaboração de Caso de Uso Impróprio

Com o intuito de percorrer as etapas de elaboração descritas, tome-se como exemplo uma simples aplicação que permita acrescentar detalhes de interações com os itens da lista de contatos de um *smart phone* Android. O principal requisito desta aplicação consiste em monitorar o uso das funções de telefone e mensagens de texto e, uma vez realizada ou recebida uma chamada telefônica, ou também enviada ou recebida uma mensagem de texto, permitir a associação de um conteúdo textual ou multimídia (som e/ou imagem) a esse evento. Esse conteúdo associado deve permanecer armazenado em um repositório específico, podendo ser acessado a partir da lista de contatos, da lista de chamadas realizadas/recebidas, ou da lista de mensagens enviadas/recebidas. Um exemplo de diagrama de caso de uso para esta aplicação é apresentado na Fig. 27.

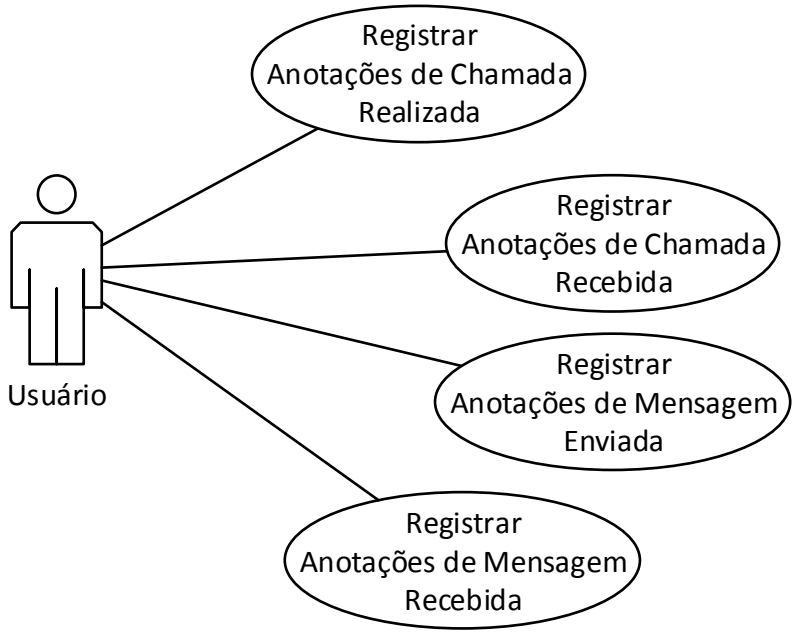


Figura 27 - Exemplo de Diagrama de Caso de Uso.

Os casos de uso de tais requisitos - em uma versão bastante simplificada - para as chamadas telefônicas são apresentados nas páginasseguintes.

UC001: Registrar anotações de chamada realizada
ATORES: Usuário
DESCRIÇÃO: Usuário anota detalhes de uma chamada telefônica realizada para um contato.
Pré-condição: Contato registrado na lista de contatos, função telefone totalmente operacional, chamada telefônica realizada (com ou sem sucesso).
Pós-condição: Anotações de chamada realizada registradas com sucesso na base de dados do sistema.
Fluxo básico: 1 – Usuário aciona registro de anotações; 2- Sistema apresenta tela para registro de anotações; 3 – Usuário preenche a tela com as anotações acerca da chamada; 4 – Usuário finaliza as anotações; 5 – Sistema associa anotações ao contato e à chamada e faz a gravação na base de dados. 6 – Sistema retorna ao estado de espera aguardando nova ativação.0

UC002: Registrar anotações de chamada recebida
ATORES: Usuário
DESCRIÇÃO: Usuário anota detalhes de uma chamada telefônica recebida de um contato.
Pré-condição: Contato registrado na lista de contatos, função telefone totalmente operacional, chamada telefônica recebida (com ou sem sucesso).
Pós-condição: Anotações de chamada recebida registradas com sucesso na base de dados do sistema.
Fluxo básico: 1 – Usuário aciona registro de anotações; 2- Sistema apresenta tela para registro de anotações; 3 – Usuário preenche a tela com as anotações acerca da chamada recebida; 4 – Usuário finaliza as anotações; 5 – Sistema associa anotações ao contato e à chamada e faz a gravação na base de dados. 6 – Sistema retorna ao estado de espera aguardando nova ativação.

Já os casos de uso para o envio e recebimento de mensagens de texto são os apresentados nas páginas a seguir.

UC003: Registrar anotações de mensagem enviada
ATORES: Usuário
DESCRIÇÃO: Usuário anota detalhes de uma mensagem enviada para um contato da lista.
Pré-condição: Contato registrado na lista de contatos, função telefone totalmente operacional, mensagem enviada.
Pós-condição: Anotações de mensagem enviada registradas com sucesso na base de dados da aplicação.
Fluxo básico: 1 – Usuário aciona registro de anotações; 2- Sistema apresenta tela para registro de anotações; 3 – Usuário preenche a tela com as anotações acerca da chamada recebida; 4 – Usuário finaliza as anotações; 5 – Sistema associa anotações ao contato e à chamada e faz a gravação na base de dados. 6 – Sistema retorna ao estado de espera aguardando nova ativação.

UC004: Registrar anotações de mensagem recebida
ATORES: Usuário, Sistema
DESCRIÇÃO: Usuário anota detalhes de uma mensagem recebida de um contato da sua lista.
Pré-condição: Contato registrado na lista de contatos, função telefone totalmente operacional, mensagem recebida.
Pós-condição: Anotações de mensagem recebida registradas com sucesso na base de dados do sistema.
Fluxo básico: 1 – Usuário aciona registro de anotações; 2- Sistema apresenta tela para registro de anotações; 3 – Usuário preenche a tela com as anotações acerca da chamada recebida; 4 – Usuário finaliza as anotações; 5 – Sistema associa anotações ao contato e à chamada e faz a gravação na base de dados. 6 – Sistema retorna ao estado de espera aguardando nova ativação.

Os casos de uso apresentados são bastante simplificados, com o único propósito de exemplificar o processo de elaboração de um caso de uso impróprio. De acordo com o processo apresentado, a primeira etapa objetiva identificar os atores. Pode-se inferir ao menos a existência do ator Usuário (malicioso ou não). O usuário dispõe do acesso ao dispositivo móvel e ao sistema, e pode representar um risco considerável ao fazer uso impróprio ou – sendo um usuário malicioso – atacar o sistema ou sua base de dados. Além disso o próprio sistema pode apresentar falhas ou erros que danificam ou invalidam os registros de anotações, as quais são avaliadas na etapa seguinte. Os diagramas de caso de uso impróprio para essas ocorrências são então elaborados, como os exemplos apresentados na Fig. 28 a seguir.

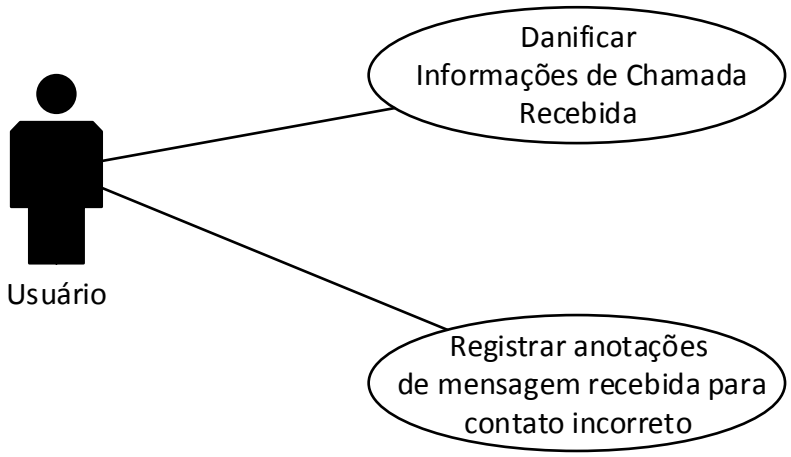


Figura 28 - Exemplo de Diagrama de Caso de Uso impróprio.

Uma vez elaborado o diagrama, os casos de uso impróprio para tais ocorrências podem ser descritos, tais como os apresentados – de modo bastante simplificado - na sequência.

MUC001: Danifica anotações de mensagem recebida
ATORES: Usuário
DESCRIÇÃO: Usuário acessa e altera de forma indiscriminada detalhes de uma mensagem recebida de um contato da sua lista.
Pré-condição: Anotação de detalhes de uma chamada recebida devidamente registrada na base de dados do sistema.
Pós-condição: Anotações de chamada recebida danificada ou incorreta na base de dados do sistema.
Fluxo básico: 1 – Usuário seleciona um contato e aciona registro de anotações de chamada; 2 – Sistema apresenta tela com o registro de anotações; 3 – Usuário provoca a mudança para o modo de edição; 4 – Usuário altera as anotações acerca da chamada recebida; 5 – Usuário confirma as alterações; 6 – Sistema associa anotações ao contato e à chamada e faz a gravação na base de dados. 7 – Sistema retorna ao estado de espera aguardando nova ativação.

Nesta ocorrência, quer seja por acidente ou intencionalmente, o usuário pode alterar informações registradas para uma chamada de modo a deixá-la inconsistente, incompleta ou inutilizável. Um segundo caso de uso impróprio é apresentado a seguir.

MUC002: Registra anotações de mensagem recebida para contato incorreto
ATORES: Usuário
DESCRIÇÃO: Usuário anota detalhes de uma mensagem recebida de um contato da sua lista, porém o Sistema a associa a outro contato.
Pré-condição: Contatos registrados na lista de contatos, função telefone totalmente operacional, mensagem recebida.
Pós-condição: Anotações de mensagem enviada registradas para contato incorreto na base de dados do sistema.
Fluxo básico: 1 – Usuário aciona registro de anotações; 2 – Sistema apresenta tela para registro de anotações; 3 – Usuário preenche a tela com as anotações acerca da chamada recebida; 4 – Usuário finaliza as anotações; 5 – Sistema associa anotações a outro contato e à chamada e faz a gravação na base de dados. 6 – Sistema retorna ao estado de espera aguardando nova ativação.

Nesta ocorrência uma falha do sistema altera a identificação do contato e as informações registradas para uma chamada são associadas a outro contato, provocando uma inconsistência.

C.2 Análise de Risco

Para demonstrar uma análise de riscos como proposto tome-se como exemplo a aplicação apresentada no item C.1.1, “Elaboração de Caso de Uso Impróprio”, utilizando o método qualitativo e considerando as seguintes variáveis: risco, probabilidade, dano e impacto. A seguir apresenta-se a compilação desta análise, abrangendo uma pequena parte do universo das informações até então produzidas acerca de falhas, faltas, ameaças e vulnerabilidades.

A Tabela 7 apresenta a análise de riscos do ambiente operacional do software, contemplando:

- A identificação do risco, por meio de um identificador único que permita a rastreabilidade durante todo o SDLC;

- Uma descrição textual do risco, que permita o entendimento e a caracterização do risco;
- O agente da ameaça, vetor por meio do qual o risco pode transformar-se em incidente;
- A vulnerabilidade, que pode ser explorada pelo agente para efetivar o ataque;
- A probabilidade de efetivação do risco, em formato de escala de Likert (com valores Muito Baixo, Baixo, Médio, Alto, Muito Alto), e com base no histórico de incidentes ou ocorrências similares;
- O impacto para a segurança da informação, caso o risco se concretize, também expresso em formato de escala de Likert (com valores Muito Baixo, Baixo, Médio, Alto, Muito Alto).

A Tabela 8 apresenta a análise de riscos dos canais de comunicação contemplando os mesmos critérios, e a Tabela 9 apresenta a análise de riscos para o ambiente de desenvolvimento. Aqui convém ressaltar que a maioria – se não a totalidade – do software desenvolvido para Android é produzido em ambiente PC, utilizando ferramentas como o *Android Studio and SDK Tools*. Uma consequência imediata é a possibilidade de que ameaças e vulnerabilidades do ambiente PC possam gerar problemas de segurança da informação para o software em desenvolvimento, embora durante as pesquisas não tenha sido identificado nenhum *malware* que faça uso desde ambiente para infectar o software desenvolvido.

Pode-se inferir que os riscos apresentados até este ponto são decorrentes de ameaças típicas dos ambientes de computação móveis e podem ser considerados genéricos, quer seja, são ofensores a praticamente todo e qualquer tipo de software característicos destes ambientes. De certo ponto de vista isto pode ser considerado normal, e reforça a necessidade de uma base de conhecimento como mencionado na Seção 4.3, tornando a reutilização da informação uma prática e permitindo o refinamento sucessivo e a melhoria contínua da segurança da informação.

Na Tabela 10 são avaliados de forma mais específica alguns riscos do software em desenvolvimento, decorrentes dos requisitos anotados na fase de especificação. É importante ressaltar que os riscos decorrentes dos requisitos podem derivar-se, seguir-se ou sobrepor-se aos riscos "genéricos" das análises anteriores, agravando ou reduzindo os impactos ou as vulnerabilidades.

Os riscos classificados a partir da probabilidade alta e do impacto alto são fortes candidatos a análises futuras, incluindo quantificação, e uma gerência de risco mais próxima e

específica, na qual a classificação de risco é estabelecida usando uma matriz e uma escala de risco apropriada para cada risco.

Tabela 7 - Riscos do Ambiente operacional.

Ambiente Operacional						
Item	**Risco**	**Ameaça**	**Vulnerabilidade**	**Probabilidade**	**Impacto**	**Comentário**
#1	Acesso às informações da base de dados	*Malware*	Comunicação inter-processos do Android	Alta	Alto	Um atacante pode contaminar uma aplicação qualquer e por meio da mesma acessar os dados do software sem conhecimento do usuário.
#2	Roubo de identidade / credenciais do usuário	*Malware*	Criptografia fraca	Baixa	Alto	Um atacante pode identificar o usuário e obter acesso às suas credenciais para acesso aos dados do sistema ou uso de funcionalidades do ambiente.
#3	Interrupção da aplicação durante operação crítica	Funcionalidade do Hardware	Alimentação – bateria	Alta	Alto	O sistema pode finalizar o software durante uma atualização do banco de dados ou transferência de dados pela rede.
#4	Interrupção da aplicação durante operação crítica	Funcionalidade do Hardware	Sobrecarga no processamento	Baixa	Alto	O sistema pode “travar” durante uma atualização do banco de dados ou transferência de dados pela rede devido ao uso excessivo da CPU.
#5	Interrupção da aplicação durante operação crítica	Funcionalidade do Hardware	Falha no gerenciamento de memória	Baixa	Alto	O sistema pode finalizar o software durante uma atualização do banco de dados ou transferência de dados pela rede.
#6	Uso impróprio dos recursos do ambiente	*Malware*	Proteção dos componentes do software	Média	Baixo	Um atacante pode comprometer componentes do software com o intuito de utilizar os recursos do ambiente computacional – como acesso à rede, envio de SMS ou execução de chamadas para números pré-definidos.

Tabela 8 - Riscos dos canais de comunicação.

Canais de comunicação						
Item	**Risco**	**Ameaça**	**Vulnerabilidade**	**Probabilidade**	**Impacto**	**Comentário**
#7	Interceptação de comunicação	Rede WiFi	Autenticação ou autorização	Média	Médio	Um atacante pode explorar a conexão do usuário a um ponto de acesso à rede sem fio devido à ausência de senha ou senha fraca no protocolo de conexão.
#8	Interceptação de comunicação	Comunicação por rádio (3G, 4G)	Autenticação ou autorização	Muito Baixa	Médio	Um atacante pode explorar a conexão do usuário a uma estação rádio base com uso de equipamento especial ou clonado
#9	Simulação ou emulação	Comunicação por rádio (3G, 4G)	Autenticação ou autorização	Muito Baixa	Alto	Um atacante pode simular a conexão do usuário a uma estação rádio base com uso de equipamento especial ou clonado.
#10	Captura de informações	Comunicação por rádio (3G, 4G)	Autenticação ou autorização	Baixa	Alto	Um atacante pode interceptar a comunicação do usuário com a estação rádio base e capturar as informações transferidas.
#11	Captura de informações	Comunicação por *Bluetooth*	Autenticação ou autorização	Média	Alto	Um atacante pode interceptar a comunicação do usuário com um dispositivo *Bluetooth* e capturar as informações transferidas ou acessar informações do ambiente.

Tabela 9 - Riscos do ambiente de desenvolvimento.

Ambiente de desenvolvimento						
Item	**Risco**	**Ameaça**	**Vulnerabilidade**	**Probabilidade**	**Impacto**	**Comentário**
#12	Manipulação de entrega de conteúdo	*Malware*	Proteção do ambiente computacional	Média	Alto	Um atacante pode comprometer o ambiente de desenvolvimento e inserir código indesejado no software para fazer Uso impróprio do equipamento do usuário, como o envio de mensagens ou a realização de ligações telefônicas.
#13	Anúncios indesejados	*Malware*	Proteção do ambiente computacional	Média	Médio	O ambiente pode favorecer ou permitir a inserção de código que facilite ou possibilite a veiculação de anúncios indesejados – *adware* – no software
#14	Manipulação de mecanismos de busca	*Malware*	Proteção do ambiente computacional	Alta	Muito Baixo	O ambiente pode favorecer ou permitir a inserção de código que simule o acesso a sites específicos com o intuito de incrementar o ranking de acesso a tais sites.
#15	Uso impróprio de recursos computacionais	Spam	Proteção do ambiente computacional	Alta	Médio	O ambiente pode favorecer ou permitir a inserção de código que facilite ou possibilite o envio de mensagens de e-mail indesejadas (*spam*).

Tabela 10 - Riscos associados aos requisitos.

Item	Requisito	Ameaça	Vulnerabilidade	Probabilidade	Impacto	Comentário
#16	Monitorar o uso das funções de telefone	*Malware*	Comunicação inter-processos do Android	Alta	Muito Alto	Um atacante pode contaminar a aplicação e acessar os dados uso do telefone sem conhecimento do usuário.
#17	Monitorar o uso das mensagens de texto	*Malware*	Comunicação inter-processos do Android	Alta	Alto	Um atacante pode contaminar a aplicação e acessar os dados uso de mensagens sem conhecimento do usuário.
#18	Permitir a associação de um conteúdo textual ao registro de ligações e mensagens	Falha	Disponibilidade dos serviços de acesso ao banco de dados	Baixa	Médio	O serviço de acesso ao banco de dados pode ficar comprometido ou inoperante, comprometendo a integridade devido à impossibilidade de uma atualização.
#19	Armazenar conteúdo associado em um repositório específico	Falha	Disponibilidade dos serviços de acesso ao banco de dados	Baixa	Médio	O serviço de acesso ao banco de dados pode ficar comprometido ou inoperante, comprometendo a integridade devido à impossibilidade de uma atualização.
#20	Acessar conteúdo associado em um repositório específico	Falha	Disponibilidade dos serviços de acesso ao banco de dados	Baixa	Médio	O serviço de acesso ao banco de dados pode ficar comprometido ou inoperante, comprometendo a disponibilidade.

C.3 Máquina de ataque

O documento a seguir é um *template* para a elaboração de uma especificação de máquina de ataque, cujas descrições e orientações para a elaboração foram apresentadas na seção 4.6.

Id		**Versão**	
Descrição			
Autor		**Data**	
Projeto			
Diagrama UML:			
Algoritmo:			
AIDD – *Attacks Identification Description and Defense*:			
Requisitos relacionados:			
Casos de uso impróprio relacionados:			

C.3.1 Modelagem de máquina de ataque

Partindo-se dos casos de uso impróprio elencados é possível elaborar os modelos das máquinas de ataque que possam reproduzir ou simular a ocorrência descrita, efetivando-se assim o risco com o intuito de comprovar a eficácia das contramedidas adotadas. Os modelos de máquina de ataque propostos apresentam as informações obtidas desde o início do processo de desenvolvimento do software, como já exposto. Estas informações são avaliadas e consolidadas em um documento único, que contém:

- A **identificação** do modelo da máquina de ataque, isto é, um identificador único para o modelo durante todo o processo, que permite a rastreabilidade do documento e a sua vinculação com os demais processos estabelecidos;
- A **versão** do documento;
- A **descrição** da máquina de ataque ou da classe de máquinas de ataque com o objetivo ou características do ataque por ela modelado;
- O registro do **autor** e da **data** do documento;
- A identificação do **projeto** para o qual a máquina de ataque se destina;
- O **diagrama UML** com a representação gráfica das classes que compõem a máquina de ataque em questão, sua hierarquia e suas interações;
- O **algoritmo** utilizado para o projeto da máquina de ataque, o qual fundamentará a construção do código de sua implementação;
- Uma análise com base no modelo **AIDD – *Attack Identification Description and Defense,*** proposto por Souissi e Serhrouchni (2014). Este modelo visa classificar e caracterizar determinados tipos (classes) de ataques e permite, deste modo, identificar características comuns que podem ser combatidas por defesas específicas. Isto favorece o projetista e o desenvolvedor na construção da máquina de ataque, e também o testador, quando da sua utilização. Além disso, o detalhamento do ataque perpetrado pela máquina de ataque ora projetada favorece a estruturação das defesas. Essa análise é composta de quatro categorias de informação:
  1) A **fonte**, isto é, a origem do ataque, que pode ser subdividida em local ou remota, ou seja, o atacante ou ofensor encontra-se no mesmo ambiente computacional do software ou em outro ambiente, acessando este por meio de uma rede;

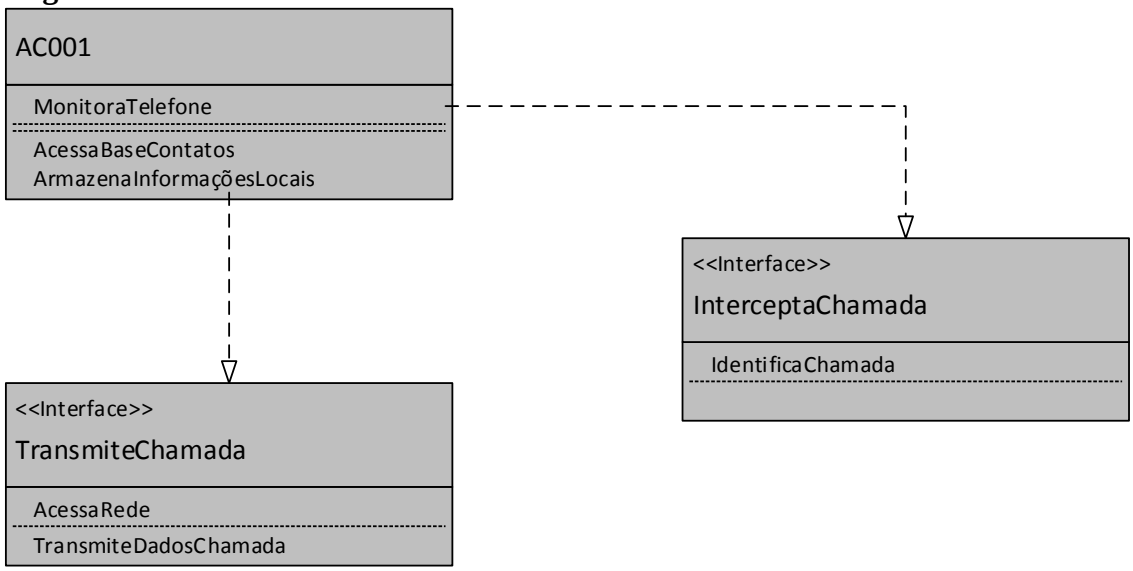
2) O **alvo** do ataque, quer seja, qual componente do ambiente computacional é visado pelo atacante. Este alvo pode ser também uma vulnerabilidade do sistema operacional, uma falha do protocolo de rede, partes, funções ou fragilidades da aplicação – do software propriamente dito, que possam ser exploradas com o intuito de possibilitar o ataque.
3) O **vetor**, ou seja, o método utilizado pelo atacante para efetivar o ataque. Pode incluir vulnerabilidades, uma vez que estas são necessárias para o sucesso do ataque, incluindo falhas de validação, existência de *exploits*, falha de projeto, *spoofing*, falhas de configuração e engenharia social, entre outros.
4) O **resultado** esperado do ataque, do ponto de vista do impacto, que pode incluir uma negação de serviço, a escalação de privilégios, a desativação de defesa, o uso de recursos específicos, a disseminação de código ou a obtenção de informações, por exemplo. Pode incluir uma falha do ataque, indicando que o mesmo não foi bem sucedido.

– Os **requisitos** ou regras de negócio que têm relação com a máquina de ataque, quer seja por implicar em uma vulnerabilidade, quer seja por serem afetados pelos ataques ou impactados pelos resultados desse ataque;
– Os **Casos de Uso impróprio** relacionados com a máquina de ataque em questão, ou seja, aqueles utilizados para a modelagem da mesma.

A seguir são apresentadas as definições de máquinas de ataque de acordo com o modelo proposto.

Id	MA001	**Versão**	1.0
Descrição	Este modelo apresenta as classes da máquina de ataque para a simulação da ocorrência de um ataque no qual o Atacante intercepta o registro de detalhes de uma chamada telefônica realizada para um contato e transmite tais informações para um servidor usando a rede.		
Autor	Luis Gonzaga de Paulo	**Data**	20/03/2015
Projeto	Sistema exemplo		

Diagrama UML:



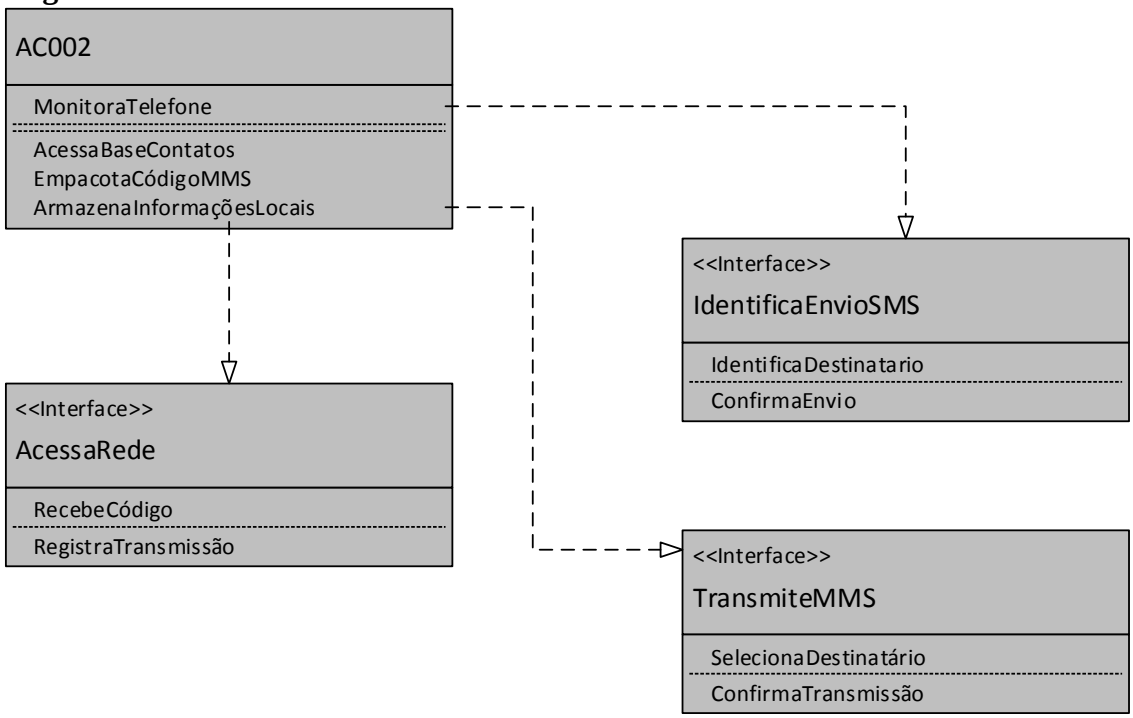
Algoritmo:

```
Faça Enquanto ligado
     Verifica Estado_Telefone
     Se Estado_Telefone = Em_Repouso
     Então verifica Chamada_Realizada
          Se Chamada_Realizada = "OK" E Registra_Chamada Não
          Contém ID_Chamada
          Então
               Captura Dados_Chamada
               Registra_Chamada Armazena ID_Chamada
          Fim-Então
          Se Rede_Disponível = "OK"
          Então
               Acessa_Servidor
               Transmite_Dados
          Fim-Então
          Senão
               Acessa_Base_Local
               Armazena Dados_Chamada
               Registra_Chamada = ""
               Agenda_Evento Transmissao_Dados
          Fim-Senão
     Fim-Então
FIM.
```

AIDD – *Attacks Identification Description and Defense*
Origem: Local – Sistema Operacional ou aplicação instalada no telefone. **Alvo:** Banco de dados da aplicação **Vetor:** Comunicação inter-processos e gerenciamento de memória do Android **Resultado:** Exposição dos dados das chamadas realizadas – violação da confidencialidade
Requisitos relacionados: RF1 e RF2
Casos de Uso impróprio relacionados: MUC001

Id	MA002	**Versão**	1.0
Descrição	Atacante detecta o envio de SMS para um destinatário da base de dados do software e, por meio da interceptação dos dados, acessa um código malicioso com o intuito de propagá-lo e faz o envio deste código em formato MMS para aquele destinatário.		
Autor	Luis Gonzaga de Paulo	**Data**	20/03/2015
Projeto	Sistema exemplo		

Diagrama UML:



Algoritmo:

```
Faça Enquanto ligado
     Verifica Estado_Telefone
     Se Estado_Telefone = Envia_SMS
     Então verifica Codigo_Disponivel
          Se Codigo_Disponivel = "OK" E Registra_SMS Não
          Contém ID_SMS
          Então
               Empacota Codigo_MMS
               Registra_SMS Armazena ID_SMS
               Acessa_Rede_Telefone
               Transmite_MMS
          Fim-Então
          Senão
               Acessa_Base_Local
               Armazena Dados_SMS
               Registra_SMS = ""
               Agenda_Evento Transmissao_MMS
          Fim-Senão
     Fim-Então
FIM.
```

AIDD – *Attacks Identification Description and Defense*
Origem: Local – Sistema Operacional ou aplicação instalada no telefone. **Alvo:** Serviço de envio de mensagens multimídia (MMS) **Vetor:** Comunicação Inter processos e gerenciamento de memória do Android **Resultado:** Envio de conteúdo não desejado. Disseminação de *malware*.
Requisitos relacionados: RF03 e RF04
Casos de Uso impróprio relacionados: MUC002

Apêndice D – Estudo de caso

Com o objetivo de avaliar o modelo SDD foi proposto o desenvolvimento de um software para Android fazendo uso do modelo. O software em questão contempla extensões de funcionalidades da agenda do *smart phone*, integrando-as com outras funcionalidades típicas do dispositivo, como, por exemplo, o envio de mensagens de texto e multimídia. Mais do que para as funcionalidades e o desempenho do software, para efeito deste trabalho, a importância aqui desloca-se para as implicações do uso do modelo em termos de técnica, esforço e tempo, e também quanto à capacidade de atender o propósito de reforçar a segurança da informação no processo de desenvolvimento.

D.1 O software µCRM

O µCRM é também um protótipo acadêmico cujo desenvolvimento tem o propósito de prestar-se a ser um estudo de caso, tratando-se da aplicação da metodologia SDD – *Security Drive Development*. Os itens a seguir apresentam o detalhamento do software.

D.1.1 Os requisitos funcionais

O software µCRM deverá atender no mínimo os seguintes requisitos funcionais:

RF1. Permitir a classificação de Contatos da agenda do dispositivo como cliente;

RF2. Manter uma base de dados própria com informações adicionais vinculadas aos Contatos da agenda do dispositivo, tais como:

- 2.1. Aniversário
- 2.2. Data de formatura
- 2.3. Profissão / formação
- 2.4. Interesses
- 2.5. Nome do Conjunge
- 2.6. Nome dos Filhos
- 2.7. Data de Casamento

RF3. Garantir que o acesso à base de dados seja possível apenas por meio do próprio software;

RF4. Registrar automaticamente os eventos a seguir, com data, hora e local (se disponível):

4.1. Registrar a realização de chamadas telefônicas para os números de Contatos identificados como cliente;
4.2. Registrar o recebimento de chamadas telefônicas para os números de Contatos identificados como cliente;
4.3. Registrar o envio de mensagens de texto para os números de Contatos identificados como cliente;
4.4. Registrar o recebimento mensagens de texto para os números de Contatos identificados como cliente;
4.5. Registrar o envio de mensagens multimídia para os números de Contatos identificados como cliente;
4.6. Registrar o recebimento mensagens multimídia para os números de Contatos identificados como cliente;

RF5. Permitir a associação de texto ao registro de eventos de Contatos identificados como cliente;
RF6. Permitir a associação de mensagem de áudio ao registro de eventos de Contatos identificados como cliente;
RF7. Permitir a associação de imagens ao registro de eventos de Contatos identificados como cliente;
RF8. Permitir a associação vídeos ao registro de eventos de Contatos identificados como cliente;
RF9. Permitir a consulta ao registro de eventos associados ao cliente por nome, número do telefone ou período (data e hora) do evento;
RF10. Permitir a exportação dos dados de eventos associados ao cliente das seguintes formas:
10.1. Em formato XML;
10.2. Em formato de algum Banco de Dados padrão – MySQL, por exemplo;
10.3. Em formato de planilha compatível com Microsoft Excel;
10.4. Encaminhada por meio de e-mail;
10.5. Permitir a seleção dos dados a serem exportados;
10.6. Permitir a definição de intervalo de datas e horário a serem exportados
RF11. Interagir com o Calendário do dispositivo, de modo a permitir o agendamento de eventos e o aviso da ocorrência de eventos;

RF12. Prover o acesso ao conteúdo associado a partir da lista de contatos, da lista de chamadas realizadas ou recebidas, ou da lista de mensagens enviadas ou recebidas;

D.1.2 Os requisitos não funcionais

O software µCRM deverá atender no mínimo os seguintes requisitos não funcionais:

RNF1. Executar em ambiente operacional Android;

RNF2. Manter uma cópia remota da base de dados;

RNF3. Prover a cópia ou exportação da base de dados para serviço de armazenagem externo (cloud, rede, etc.);

RNF4. Criptografar a base de dados;

D.1.3 O diagrama de casos de uso

O diagrama de casos de uso da Fig.29 representa as funcionalidades elencadas para o software, contemplando o ator e as interações expressas na forma gráfica, de acordo com o padrão UML.

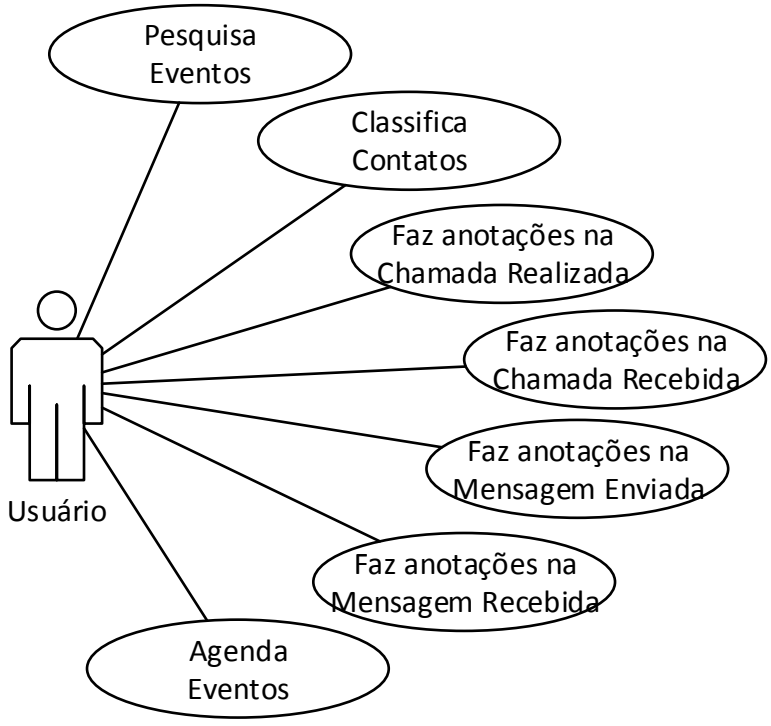


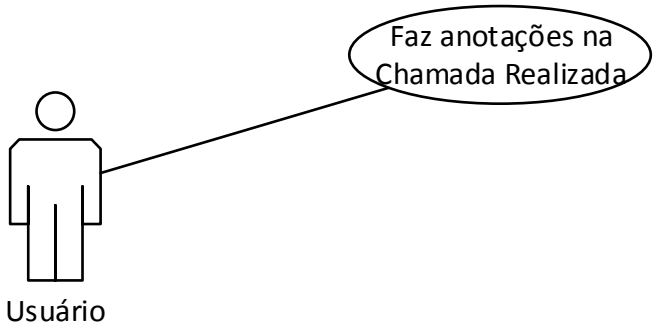
Figura 29 - Diagrama de casos de uso

D.1.4 Os casos de uso

Os casos de uso do sistema µCRM foram elaborados a partir dos requisitos do sistema, e detalham as funcionalidades que compõem o sistema, apresentando as condições e a sequência para a realização de uma operação por meio destas funcionalidades. Os casos de uso também fazem referência aos requisitos de modo a manter a rastreabilidade. A seguir são apresentados os principais Casos de Uso do Sistema µCRM.

Id	UC001	**Versão**	1.0
Projeto	µCRM – Sistema de CRM para Android		
Autor	Luis Gonzaga de Paulo	**Data**	06/03/2015
Descrição	Usuário anota detalhes de uma chamada telefônica realizada para um contato.		
Atores	Usuário		

Diagrama de caso de uso:



Pré-condição:
Contato registrado na lista de contatos
Função telefone totalmente operacional
Chamada telefônica realizada (com ou sem sucesso)

Pós-condição:
Anotações de chamada realizada registradas com sucesso na base de dados do sistema.

Fluxo básico:

- fb1: Usuário aciona registro de anotações;
- fb2: Sistema apresenta tela para registro de anotações;
- fb3: Usuário preenche a tela com as anotações acerca da chamada;
- fb4: Usuário finaliza as anotações;
- fb5: Sistema associa anotações ao contato e à chamada e faz a gravação na base de dados.
- fb6: Sistema retorna ao estado de espera aguardando nova ativação.

Fluxo alternativo:
N/A

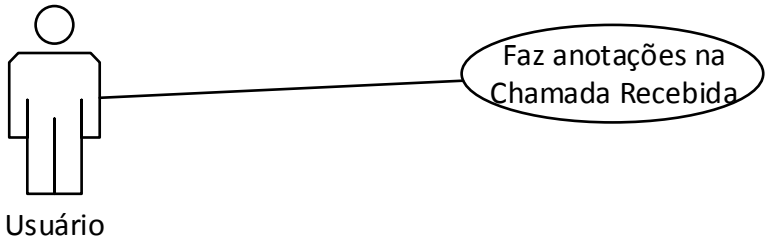
Exceções:

- ex1: Sistema indisponível: sistema informa usuário sobre problema e finaliza a execução.

Requisitos relacionados:
RF4.1
RF5
RF6
RF7
RF8
RNF4

Id	UC002	**Versão**	1.0
Projeto	µCRM – Sistema de CRM para Android		
Autor	Luis Gonzaga de Paulo	**Data**	06/03/2015
Descrição	Usuário anota detalhes de uma chamada telefônica recebida de um contato.		
Atores	Usuário		

Diagrama de caso de uso:



Pré-condição:
Contato registrado na lista de contatos
Função telefone totalmente operacional
Chamada telefônica recebida (com ou sem sucesso)

Pós-condição:
Anotações de chamada recebida registradas com sucesso na base de dados do sistema.

Fluxo básico:

fb1: Usuário aciona registro de anotações;
fb2: Sistema apresenta tela para registro de anotações;
fb3: Usuário preenche a tela com as anotações acerca da chamada recebida;
fb4: Usuário finaliza as anotações;
fb5: Sistema associa anotações ao contato e à chamada e faz a gravação na base de dados.
fb6: Sistema retorna ao estado de espera aguardando nova ativação.

Fluxo alternativo:
N/A

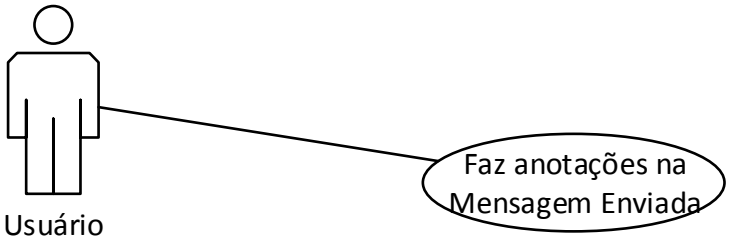
Exceções:

ex1: Sistema indisponível: sistema informa usuário sobre problema e finaliza a execução.

Requisitos relacionados:
RF4.2; RF5; RF6; RF7; RF8; RNF4

Id	UC003	**Versão**	1.0
Projeto	µCRM – Sistema de CRM para Android		
Autor	Luis Gonzaga de Paulo	**Data**	06/03/2015
Descrição	Usuário anota detalhes de uma mensagem enviada para um contato da lista.		
Atores	Usuário		

Diagrama de caso de uso:



Pré-condição:
Contato registrado na lista de contatos
Função telefone totalmente operacional
Mensagem SMS enviada

Pós-condição:
Anotações de mensagem enviada registradas com sucesso na base de dados da aplicação.

Fluxo básico:

- fb1: Usuário aciona registro de anotações;
- fb2: Sistema apresenta tela para registro de anotações;
- fb3: Usuário preenche a tela com as anotações acerca da chamada recebida;
- fb4: Usuário finaliza as anotações;
- fb5: Sistema associa anotações ao contato e à chamada e faz a gravação na base de dados.
- fb6: Sistema retorna ao estado de espera aguardando nova ativação.

Fluxo alternativo:
N/A

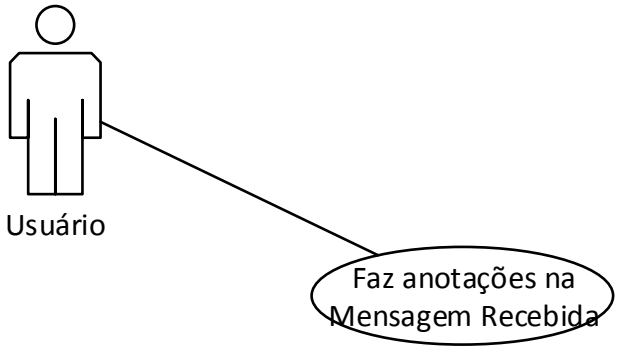
Exceções:

- ex1: Sistema indisponível: sistema informa usuário sobre problema e finaliza a execução.

Requisitos relacionados:
RF4.3; RF5; RF6; RF7; RF8; RNF4

Id	UC004	**Versão**	1.0
Projeto	µCRM – Sistema de CRM para Android		
Autor	Luis Gonzaga de Paulo	**Data**	06/03/2015
Descrição	Usuário anota detalhes de uma mensagem recebida de um contato da lista.		
Atores	Usuário		

Diagrama de caso de uso:



Pré-condição:
Contato registrado na lista de contatos identificado como cliente, função telefone totalmente operacional, mensagem recebida.

Pós-condição:
Anotações de mensagem recebida registradas com sucesso na base de dados do sistema.

Fluxo básico:

- fb1: Usuário aciona registro de anotações;
- fb2: Sistema apresenta tela para registro de anotações;
- fb3: Usuário preenche a tela com as anotações acerca da chamada recebida;
- fb4: Usuário finaliza as anotações;
- fb5: Sistema associa anotações ao contato e à chamada e faz a gravação na base de dados.
- fb6: Sistema retorna ao estado de espera aguardando nova ativação.

Fluxo alternativo:
N/A

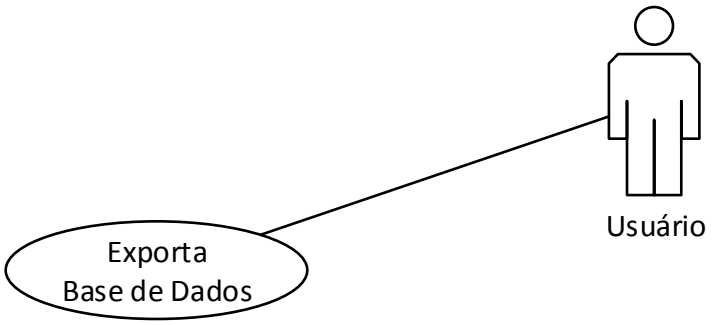
Exceções:

- ex1: Sistema indisponível: sistema informa usuário sobre problema e finaliza a execução.

Requisitos relacionados:
RF4.4; RF5; RF6; RF7; RF8; RNF4

Id	UC005	**Versão**	1.0
Projeto	µCRM – Sistema de CRM para Android		
Autor	Luis Gonzaga de Paulo	**Data**	06/03/2015
Descrição	Usuário exporta a base de dados.		
Atores	Usuário		

Diagrama de caso de uso:



Pré-condição:
Sistema em execução, telefone em estado de espera.

Pós-condição:
Base de dados gravada no local destino.

Fluxo básico:

fb1: Usuário seleciona opção exportação de dados;
fb2: Sistema apresenta tela de parametrização
fb3: Usuário preenche a tela com e informa os parâmetros da exportação;
fb4: Usuário seleciona o destino da exportação;
fb5: Sistema gera base de dados no destino de acordo com os parâmetros;
fb6: Sistema retorna ao estado de espera aguardando nova ativação.

Fluxo alternativo:

fa1: Usuário seleciona opção exportação de dados;
fa2: Sistema apresenta tela de parametrização
fa3: Usuário preenche a tela com e informa os parâmetros da exportação;
fa4: Usuário seleciona o destino da exportação;
fa5: Comunicação com o destino indisponível;
fa6: Sistema agenda exportação da base de dados para o destino de acordo com os parâmetros;
fa7: Sistema retorna ao estado de espera aguardando nova ativação.

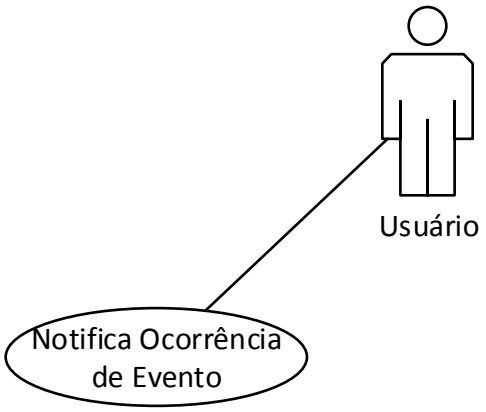
Exceções:

ex1: Sistema indisponível: sistema informa usuário sobre problema e finaliza a execução.
ex2: Espaço insuficiente para exportação da base de dados: sistema informa usuário sobre problema e solicita novo destino.

Requisitos relacionados:
RNF2; RNF3; RNF4

Id	UC006	**Versão**	1.0
Projeto	µCRM – Sistema de CRM para Android		
Autor	Luis Gonzaga de Paulo	**Data**	06/03/2015
Descrição	Usuário é notificado sobre a ocorrência de eventos.		
Atores	Usuário		

Diagrama de caso de uso:



Pré-condição:
Sistema em execução, telefone em estado de espera.

Pós-condição:
Notificação assinalada como entregue e visualizada.

Fluxo básico:

fb1: Sistema apresenta mensagem ao usuário informando evento;
fb2: Usuário confirma o recebimento da notificação;
fb3: Sistema atualiza estado da notificação para "recebida";
fb4: Sistema retorna ao estado de espera.

Fluxo alternativo:

fa1: Sistema apresenta mensagem ao usuário informando evento;
fa2: Usuário recusa ou cancela o recebimento da notificação;
fa3: Sistema agenda nova apresentação para intervalo pré-definido;
fa4: Sistema retorna ao estado de espera aguardando nova ativação.

Exceções:

ex1: Sistema indisponível: sistema informa usuário sobre problema e finaliza a execução.
ex2: Função telefone em uso: sistema aguarda função telefone retornar para estado de espera.

Requisitos relacionados:
RF4

D.1.5 O modelo de dados

O modelo de dados do sistema µCRM é bastante simples, comportando apenas cinco tabelas, como mostrado na Fig. 30.

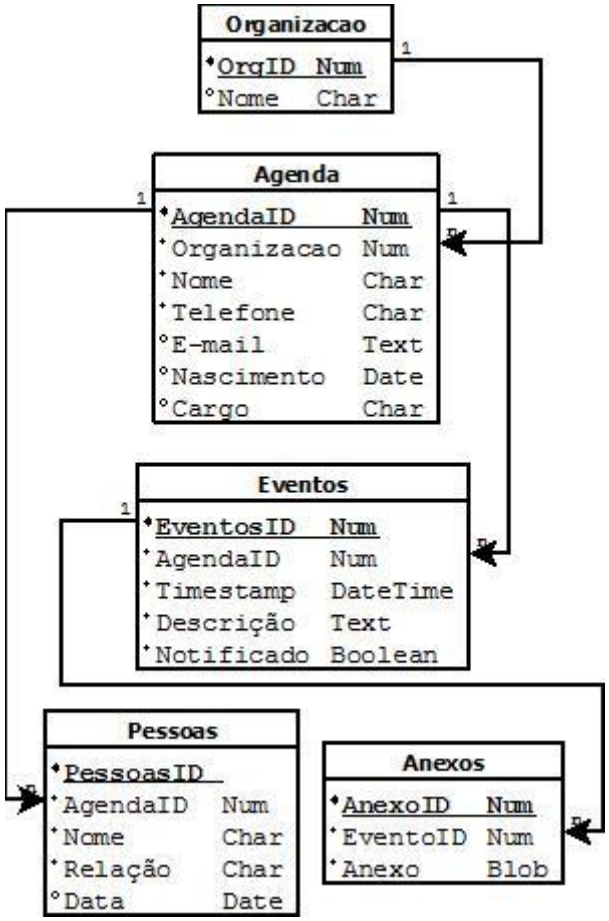


Figura 30 - Modelo de dados.

D.1.6 O dicionário de dados

Tabela Organização: Destina-se a registrar os dados da empresa, instituição ou organização à qual está vinculado um contato da agenda. É composta de:

- **Organizacao.OrgID**: Chave primária. Dado numérico, inteiro, sequencial e único que identifica uma organização em toda a base de dados.
- **Organizacao.Nome**: Cadeia de até 255 caracteres alfanuméricos que contém o nome da organização.

Tabela Agenda: Destina-se a registrar os dados do contato e é, a princípio, a agenda do próprio dispositivo – principalmente em se tratando de um *smart phone*. Caso não contemple a possibilidade de expansão ou configuração de novas colunas, deve-se considerar a criação de uma tabela auxiliar / complementar à aquela para evitar redundância. É composta de:

- **Agenda.AgendaID**: Chave primária. Dado numérico, inteiro, sequencial e único que identifica um contato em toda a base de dados.
- **Agenda.Organização**: Chave estrangeira relacionada à coluna **OrgID** da Tabela **Organizacao**.
- **Agenda.Nome**: Cadeia de até 255 caracteres alfanuméricos que contém o nome do contato.
- **Agenda.Telefone**: Cadeia de até 25 caracteres alfanuméricos que contém o telefone principal do contato.
- **Agenda.E-mail**: Bloco de texto que contém o(s) e-mail(s) do contato.
- **Agenda.Nascimento**: Data do nascimento do contato no formato DD/MM/AAAA.
- **Agenda.Cargo**: Cadeia de até 255 caracteres alfanuméricos que contém o cargo exercido pelo contato em **Organizacao**.

Tabela Eventos: Contém o registro de interações havidas com o **Contato**, por meio de ligações telefônicas iniciadas e recebidas, envio e recebimentos de mensagens de texto (SMS) ou multimídia (MMS). É composta de:

- **Eventos.EventosID**: Chave primária. Dado numérico, inteiro, sequencial e único que identifica um evento único em toda a base de dados.
- **Eventos.AgendaID**: Chave estrangeira relacionada à coluna **AgendaID** da Tabela **Agenda**.
- **Eventos.TimeStamp**: Dado numérico representando a data e a hora até em décimos de segundo na qual ocorreu o evento.
- **Eventos.Descricao**: Bloco de texto contendo a descrição ou características do evento.
- **Eventos.Notificacao**: Bit que informa o status da notificação do evento para o usuário ou sistema, sendo 0 (zero) = **NÃO NOTIFICADO** e 1 (um) = **NOTIFICADO**.

Tabela Pessoas: Contém o registro de pessoas que possuem relação com o **Contato** registrado na **Agenda**. É composta de:

- **Pessoas.PessoasID**: Chave primária. Dado numérico, inteiro, sequencial e único que identifica uma pessoa única em toda a base de dados.
- **Pessoas.AgendaID**: Chave estrangeira relacionada à coluna **AgendaID** da Tabela **Agenda**.

- **Pessoas.Nome**: Cadeia de até 255 caracteres alfanuméricos que contém o nome da pessoa.
- **Pessoas.Relacao**: Cadeia de até 255 caracteres alfanuméricos que explicita a relação de parentesco – ou outra – com o **Contato**.
- **Pessoas.Data**: Data do nascimento da pessoa no formato DD/MM/AAAA.

Tabela Anexos: Contém o complemento de informações dos eventos, podendo ser um texto de SMS, uma mensagem MMS, uma referência à um e-mail, imagens, sons ou vídeo. É composta de:

- **Anexos.AnexoID**: Chave primária. Dado numérico, inteiro, sequencial e único que identifica um anexo único em toda a base de dados.
- **Anexos.EventoID**: Chave estrangeira relacionada à coluna **EventosID** da Tabela **Eventos**.
- **Anexos.Blob**: Bloco binário

D.2 A declaração do Nível de Segurança Requerido

O software µCRM é um software que trata informações de caráter privado e sigiloso, sensíveis ao acesso, à modificação e à divulgação. Estas informações estão sob a proteção da Constituição Federal, que dispõe em seu artigo 5º, inciso XII, da Carta de 1988, ser inviolável o sigilo da correspondência e das comunicações telegráficas, de dados e das comunicações telefônicas.

Portanto, é indispensável que o software adote contramedidas ao acesso indevido ou não autorizado às informações de seu domínio, assim como torne o usuário ciente dos riscos no caso da impossibilidade de prover a segurança da informação de modo adequado.

D.3 A base de conhecimento

A base de conhecimento para atender o projeto do software µCRM é composta das informações apuradas no Cap.2, com ênfase no funcionamento e nas vulnerabilidades do sistema operacional Android e na identificação e classificação das ameaças, incluindo-se aí os *malwares.*

Por tratar-se de um modelo acadêmico e com funcionalidades simples, a opção para a manutenção desta base foi a criação de um simples repositório de documentos – a

maioria em formato texto, pdf ou html, com a identificação e versionamento dos documentos obtidos, e a criação de um índice por meio da ferramenta Google Desktop.

Além das fontes de informação relacionadas na Seção 4.3.1, foram consultadas:

- Samsung Knox blog, em https://www.samsungknox.com/en/blog-tags/android-security;
- O portal do projeto OWASP Mobile Security Project, disponível em https://www.owasp.org/index.php/OWASP_Mobile_Security_Project;
- O portal do projeto Android Open Source Project, disponível em https://source.android.com/devices/tech/security/;
- O portal do site Android Developers, em http://developer.android.com/training/articles/security-tips.html;

Estas consultas possibilitaram uma atualização das informações da base de conhecimento do projeto do software µCRM.

D.4 Os casos de uso impróprio

A etapa de especificação do software µCRM inclui a criação de casos de uso impróprio que possam fazer frente aos riscos e vulnerabilidades e contemplar a especificação de detalhes das ameaças identificadas e caracterizadas como relevantes pela análise de riscos. Os casos de uso impróprio aqui apresentados são adaptados dos modelos propostos de McDermott e Fox (1999) e de (Hartong et al, 2009), contemplando as seguintes informações:

- **ID**: Um identificador único que possibilita a referência ao caso de uso impróprio em todo o processo de desenvolvimento do software
- **VERSÃO**: Número sequencial que evidencia o registro das revisões do caso de uso impróprio.
- **AUTOR:** A identificação do responsável pela elaboração ou revisão do documento.
- **DATA**: Registro cronológico da elaboração ou última revisão do documento.
- **PROJETO**: Nome do projeto de software em desenvolvimento.
- **ATORES**: Identificação dos atores do caso de uso impróprio conforme modelo UML.
- **DIAGRAMA**: Representação gráfica do caso de uso impróprio.

- **FLUXO BÁSICO**: Sequência de operações ou atividades para concluir o caso de uso impróprio de maneira bem-sucedida.
- **FLUXO ALTERNATIVO**: Sequência de operações ou atividades para concluir o caso de uso impróprio de maneira diversa do fluxo básico, podendo ser bem-sucedida ou não.
- **EXCEÇÕES**: Condições adversas do caso de uso impróprio que podem resultar em erro ou interrupção dos fluxos.
- **PRÉ-CONDIÇÃO**: Estado do sistema ao iniciar as atividades do caso de uso impróprio.
- **PÓS-CONDIÇÃO**: Estado do sistema ao finalizar – ou interromper - as atividades do caso de uso impróprio.
- **PONTOS DE CAPTURA**: Restrições ou eventos que permitem a identificação das atividades do caso de uso impróprio e sua interrupção, prevenção ou detecção.
- **PONTOS DE EXTENSÃO**: Condições ou situações que favorecem ou potencializam o sucesso do caso de uso impróprio.
- **REQUISITOS RELACIONADOS**: Requisitos ou regras de negócio que tem relação ou são afetados pelo caso de uso impróprio.
- **CARACTERÍSTICAS DO ATACANTE**: Detalhes relativos ao atacante ou ofensor – devidamente avaliado no estudo e classificação das ameaças e na análise de riscos – que permitem a identificação, classificação e avaliação do comportamento, bem como a previsão da atuação do mesmo.
- **INTERESSADOS E RISCO**: Detalhes dos atores e demais interessados ou afetados pela execução do caso de uso impróprio e das possíveis consequências no que se refere à segurança da informação.

A adaptação efetuada o para o presente trabalho busca possibilitar a rastreabilidade para trás e para frente, de forma a ressaltar a abrangência do método por todo o ciclo de vida do processo de desenvolvimento do software. Esses casos de uso impróprio foram selecionados para exemplificar o modelo, cujo diagrama é apresentado na Fig. 31. Em seguida são apresentados alguns dos Casos de Uso impróprio do Sistema µCRM.

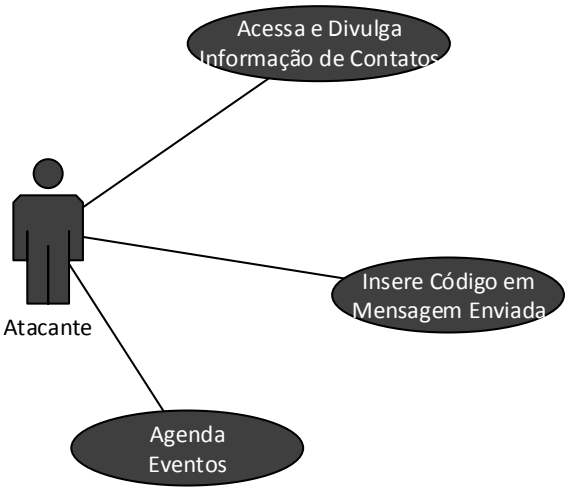
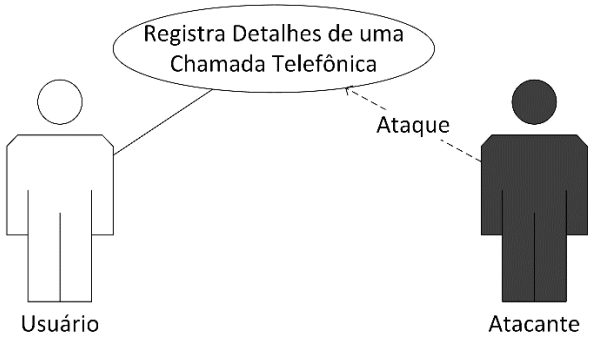


Figura 31 - Diagrama de caso de uso impróprio.

Id	MUC001	**Versão**	1.0
Descrição	Atacante intercepta o registro de detalhes de uma chamada telefônica realizada para um contato, podendo acessar e divulgar informações do contato		
Autor	Luis Gonzaga de Paulo	**Data**	20/03/2015
Projeto	µCRM – Sistema Micro CRM		
Atores	Usuário, Atacante		

Diagrama:



Fluxo básico:

- fb1: Usuário aciona registro de anotações;
- fb2: Sistema apresenta tela para registro de anotações;
- fb3: Usuário preenche a tela com as anotações acerca da chamada;
- fb4: Usuário finaliza as anotações;
- fb5: Sistema associa anotações ao contato e à chamada e faz a gravação na base de dados.
- fb6: Atacante identifica a atividade do registro na base de dados.
- fb7: Atacante acessa a base de dados com credenciais obtidas indevidamente.
- fb8: Atacante captura o registro efetivado.
- fb9: Atacante transmite informações por meio da rede.
- fb10: Sistema retorna ao estado de espera aguardando nova ativação.

Fluxo alternativo:

- fa1: (Substitui fb9) Atacante identifica rede não disponível.
- fa2: Atacante armazena informações localmente para posterior transmissão.

Exceções:

- ex1: Sistema indisponível: sistema informa usuário sobre problema e finaliza a execução.
- ex2: Atacante não consegue acesso a base de dados

Pré-condição:

Contato registrado na lista de contatos;
Função telefone totalmente operacional;
Chamada telefônica realizada (com ou sem sucesso).

Pós-condição:

Anotações de chamada realizada registradas com sucesso na base de dados do sistema. Informação capturada pelo atacante armazenada em repositório local e transmitida pela rede.

Pontos de captura:

Tentativa de acesso à rede, acesso concorrente ao banco de dados

Pontos de extensão:

Criptografia fraca, falha no controle de acesso ao banco de dados

Requisitos relacionados:

RF2; RF4.3; RF4.4; RF4.5; RF4.6; RNF4

Características do atacante: Vírus, cavalo de troia ou *spyware* com capacidade de explorar vulnerabilidades inter processos do Android ou acesso à memória compartilhada.
Interessados e Risco: Usuário do dispositivo e seus contatos podem agir como disseminadores de *malwares* e favorecer a captura de suas informações privadas.

Id	MUC002	**Versão**	1.0
Descrição	Atacante intercepta transmissão de SMS para envio de código por MMS.		
Autor	Luis Gonzaga de Paulo	**Data**	20/03/2015
Projeto	µCRM – Sistema Micro CRM		
Atores	Usuário, Atacante		

Diagrama de Caso de Uso impróprio:

Fluxo básico:

fb1: Usuário acessa base de dados de contatos;
fb2: Usuário seleciona opção "Envio de SMS";
fb3: Usuário escreve texto da mensagem;
fb4: Usuário finaliza as anotações;
fb5: Sistema envia SMS para o número telefônico do contato.
fb6: Atacante identifica a atividade de envio de SMS.
fb7: Atacante captura o número telefônico do contato.
fb8: Atacante acessa o código por meio da rede de dados.
fb9: Atacante empacota código em formato MMS.
fb10: Atacante envia MMS para o número telefônico do contato.
fb11: Sistema retorna ao estado de espera aguardando nova ativação.

Fluxo alternativo:

fa1: (Substitui fb8) Atacante identifica rede não disponível.
fa2: Atacante armazena informações localmente para tentativa posterior.

Exceções:

ex1: Sistema indisponível: sistema informa usuário sobre problema e finaliza a execução.
ex2: Atacante não consegue capturar o número telefônico do contato.
ex3: Transmissão de MMS é bloqueada pela operadora do usuário.

Pré-condição:

Contato registrado na lista de contatos;
Função telefone totalmente operacional;
Rede de dados disponível.

Pós-condição:

Envio de mensagem SMS registrada com sucesso na base de dados do sistema. Envio de código MMS confirmado pelo atacante.

Pontos de Captura:

Ativação do serviço de SMS, acesso à rede de telefonia

Pontos de extensão:

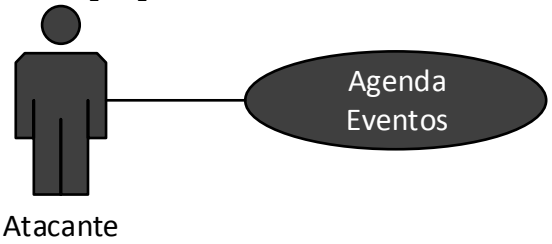
Atividade Inter processos

Requisitos relacionados:

RF3 RF4.3 RF4.4 RF4.5 RF4.6 RNF4
Características do atacante: Vírus, cavalo de troia ou *spyware* com capacidade de utilizar serviços do telefone e explorar vulnerabilidades inter processos do Android ou acesso à memória compartilhada.
Interessados e Risco: Usuário do dispositivo e seus contatos podem agir como disseminadores de *malwares*.

Id	MUC003	**Versão**	1.0
Descrição	Atacante agenda eventos para execução de atividade indevida		
Autor	Luis Gonzaga de Paulo	**Data**	20/03/2015
Projeto	µCRM – Sistema Micro CRM		
Atores	Usuário, Sistema, Atacante		

Diagrama de Caso de Uso impróprio:



Fluxo básico:

fb1: Atacante tenta executar atividade indevida.
fb2: Atacante identifica indisponibilidade de recurso (rede, função telefone, etc).
fb3: Atacante acessa agendamento de tarefas do Android
fb4: Atacante insere evento para execução posterior
fb5: Atacante retorna ao estado de espera aguardando nova ativação.

Fluxo alternativo:

fa1: (substitui fb2) Atacante identifica recurso disponível.
fa2: Atacante realiza atividade indevida com sucesso.
fa3: Atacante retorna ao estado de espera aguardando nova ativação.

Exceções:

ex1: Atacante não consegue acessar o agendamento.
ex2: Atacante não consegue inserir evento para execução posterior.

Pré-condição:
Dispositivo operando normalmente

Pós-condição:
Evento do atacante agendado

Pontos de Captura:
Tentativa de acesso à recurso do dispositivo, tentativa de acesso ao agendamento de tarefas

Pontos de extensão:
Atividade inter processos do Android, compartilhamento de memória

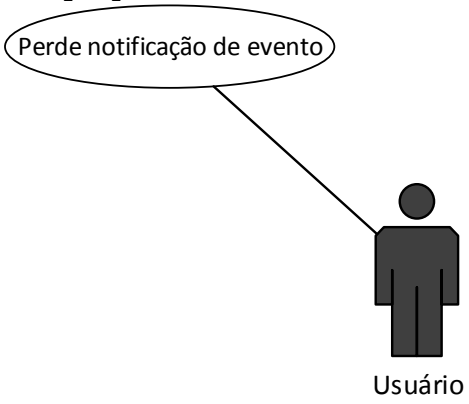
Requisitos relacionados:
RF11

Características do atacante:
Vírus, cavalo de troia ou *spyware* com capacidade de utilizar serviços do telefone e explorar vulnerabilidades inter-processos do Android ou acesso à memória compartilhada.

Interessados e Risco:
Usuário do dispositivo e seus contatos podem agir como disseminadores de *malwares*.

Id	MUC004	**Versão**	1.0
Descrição	Usuário perde a notificação de ocorrência de evento		
Autor	Luis Gonzaga de Paulo	**Data**	20/03/2015
Projeto	µCRM – Sistema Micro CRM		
Atores	Usuário		

Diagrama de Caso de Uso impróprio:



Fluxo básico:
fb1: Usuário ativa evento monitorado pelo sistema.
fb2: Sistema ignora evento ativado.
fb3: Usuário perde notificação de ocorrência de evento.
fb4: Sistema retorna ao estado de espera aguardando nova ativação.

Fluxo alternativo:
fa1: (Substitui fb2) Sistema identifica evento ativado.
fa2: Sistema registra evento com sucesso.
fa3: Usuário perde notificação de ocorrência de evento.
fa4: Sistema retorna ao estado de espera aguardando nova ativação.

Exceções:
ex1: Sistema trava

Pré-condição:
Usuário utilizando evento monitorado pelo sistema

Pós-condição:
Evento não registrado pelo sistema. Usuário sem notificação de evento.

Pontos de Captura:
N/A

Pontos de extensão:
N/A

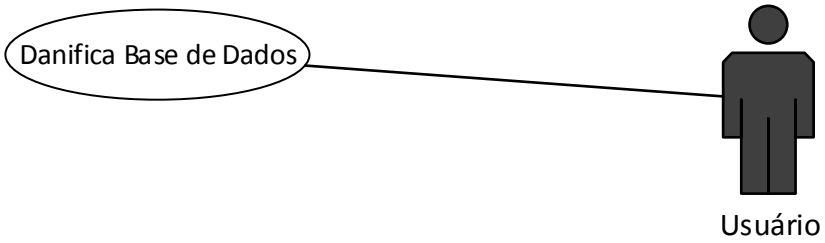
Requisitos relacionados:
RF4

Características do atacante:
Falha de processamento ou memória devido à sobrecarga, encerramento de processo ou interferência inter processo no Android.

Interessados e Risco:
Usuário perde informações de evento.

Id	MUC005	**Versão**	1.0
Descrição	Registro de evento danifica a base de dados		
Autor	Luis Gonzaga de Paulo	**Data**	20/03/2015
Projeto	µCRM – Sistema Micro CRM		
Atores	Usuário		

Diagrama de Caso de Uso impróprio:



Fluxo básico:

fb1: Usuário realiza evento identificado como registrável.
fb2: Sistema inicia atividade de atualização da base de dados.
fb3: Sistema falha na atualização.
fb4: Sistema informa falha ao usuário.
fb5: Sistema retorna ao estado de espera aguardando nova ativação.

Fluxo alternativo:

fa1: (Substitui fb4) Sistema repete o procedimento de atualização da base de dados.
fa2: Sistema retorna ao estado de espera aguardando nova ativação.

Exceções:

ex1: Sistema trava
ex2: Sistema apresenta mensagem de erro

Pré-condição:
Usuário executa atividade monitorada pelo sistema

Pós-condição:
Evento não registrado pelo sistema na base de dados

Pontos de Captura:
N/A

Pontos de extensão:
N/A

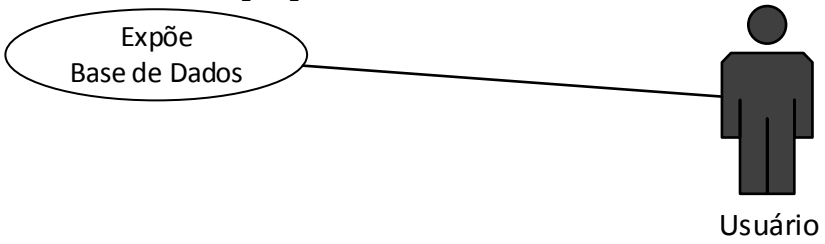
Requisitos relacionados:
RF2; RF3; RF4

Características do atacante:
Interferência no processo de acesso à base de dados

Interessados e Risco:
Usuário perde informações de eventos. Base de dados inacessível.

Id	MUC006	**Versão**	1.0
Descrição	Usuário faz acesso indevido à base de dados		
Autor	Luis Gonzaga de Paulo	**Data**	20/03/2015
Projeto	µCRM – Sistema Micro CRM		
Atores	Usuário		

Diagrama de Caso de Uso impróprio:



Fluxo básico:

fb1: Usuário inicia atividade de acesso à base de dados.
fb2: Sistema falha na autenticação de usuário.
fb3: Usuário não autenticado acessa a base de dados.
fb4: Sistema retorna ao estado de espera aguardando nova ativação.

Fluxo alternativo:

fa1: (Substitui fb2) Sistema procede a autenticação de usuário indevido.
fa2: Usuário não autenticado acessa a base de dados.
fa3: Sistema retorna ao estado de espera aguardando nova ativação.

Exceções:

ex1: Sistema trava
ex2: Sistema apresenta mensagem de erro

Pré-condição:

Usuário executa atividade que requer acesso à base de dados

Pós-condição:

Informações da base de dados acessadas indevidamente

Pontos de Captura:

Requisição de acesso à base de dados

Pontos de extensão:

Falha nos processos de criptografia/descriptografia. Chave de acesso frágil.

Requisitos relacionados:

RF2; RF3; RF4; RF12; RNF4

Características do atacante:

Interferência no processo de acesso à base de dados

Interessados e Risco:

Usuário perde informações de eventos. Base de dados inacessível.

D.5 A análise de risco

Para a análise de riscos do Sistema µCRM foi utilizado o método qualitativo que levou em consideração as seguintes variáveis: risco, probabilidade, dano e impacto. A seguir apresenta-se a compilação desta análise, ressaltando-se que, para fins desse trabalho, a análise realizada foi limitada, não abrangendo nem tampouco explorando ao máximo o universo das informações até então produzidas acerca de falhas, faltas, ameaças e vulnerabilidades. São utilizadas duas modalidades, sendo uma genérica, que leva em conta os aspectos do ambiente operacional, apresentada no Quadro 5, dos canais de comunicação e do ambiente de desenvolvimento, apresentada nos Quadros 6 e 7, e outra que contrapõe os requisitos do software às ameaças, apresentada no Quadro 8.

Quadro 5 - Riscos do Ambiente operacional.

Ambiente Operacional						
Item	**Risco**	**Ameaça**	**Vulnerabilidade**	**Probabilidade**	**Impacto**	**Comentário**
1	Acesso às informações da base de dados	Vírus ou similar	Comunicação inter-processos do Android	Alta	Alto	Um atacante pode contaminar uma aplicação qualquer e por meio da mesma acessar os dados do software sem conhecimento do usuário.
2	Roubo de identidade / credenciais do usuário	Vírus ou similar	Falha na criptografia	Baixa	Alto	Um atacante pode identificar o usuário e obter acesso às suas credenciais para acesso ao dados do sistema ou uso de funcionalidades do ambiente.
3	Interrupção da aplicação durante operação crítica	Funcionalidade do Hardware	Alimentação – bateria	Alta	Alto	O sistema pode finalizar o software durante uma atualização do banco de dados ou transferência de dados pela rede.
4	Interrupção da aplicação durante operação crítica	Funcionalidade do Hardware	Sobrecarga no processamento	Baixa	Alto	O sistema pode “travar” durante uma atualização do banco de dados ou transferência de dados pela rede devido ao uso excessivo da CPU.
5	Interrupção da aplicação durante operação crítica	Funcionalidade do Hardware	Falha no gerenciamento de memória	Alta	Alto	O sistema pode finalizar o software durante uma atualização do banco de dados ou transferência de dados pela rede.
6	Uso indevido dos recursos do ambiente	Vírus ou similar	Proteção dos componentes do software	Média	Baixo	Um atacante pode comprometer componentes do software com o intuito de utilizar os recursos do ambiente computacional – como acesso à rede, envio de SMS ou execução de chamadas para números pré-definidos
7	Inconsistência nas informações do banco de dados	Serviço do banco de dados	Disponibilidade dos serviços de acesso ao banco de dados	Baixa	Alto	O serviço de acesso ao banco de dados pode ficar comprometido ou inoperante, comprometendo a integridade devido à impossibilidade de uma atualização.

Quadro 6 - Riscos dos canais de comunicação.

Canais de comunicação						
Item	**Risco**	**Ameaça**	**Vulnerabilidade**	**Probabilidade**	**Impacto**	**Comentário**
8	Interceptação de comunicação	Rede WiFi	Autenticação ou autorização	Média	Médio	Um atacante pode explorar a conexão do usuário a um ponto de acesso à rede sem fio devido a ausência de senha ou senha fraca no protocolo de conexão.
9	Interceptação de comunicação	Comunicação por rádio (3G, 4G)	Autenticação ou autorização	Baixa	Médio	Um atacante pode explorar a conexão do usuário a uma estação rádio base com uso de equipamento especial ou clonado
10	Simulação ou emulação	Comunicação por rádio (3G, 4G)	Autenticação ou autorização	Baixa	Alto	Um atacante pode simular a conexão do usuário a uma estação rádio base com uso de equipamento especial ou clonado.
11	Captura de informações	Comunicação por rádio (3G, 4G)	Autenticação ou autorização	Baixa	Alto	Um atacante pode interceptar a comunicação do usuário com a estação rádio base e capturar as informações transferidas.
12	Captura de informações	Comunicação por *bluetooth*	Autenticação ou autorização	Baixa	Alto	Um atacante pode interceptar a comunicação do usuário com um dispositivo *bluetooth* e capturar as informações transferidas ou acessar informações do ambiente.

Quadro 7 - Riscos do ambiente de desenvolvimento.

Ambiente de desenvolvimento						
Item	**Risco**	**Ameaça**	**Vulnerabilidade**	**Probabilidade**	**Impacto**	**Comentário**
13	Manipulação de entrega de conteúdo	Vírus ou similar	Proteção do ambiente computacional	Média	Alto	Um atacante pode comprometer o ambiente de desenvolvimento e inserir código indesejado no software para fazer uso indevido do equipamento do usuário, como o envio de mensagens ou a realização de ligações telefônicas.
14	Anúncios indesejados	Vírus ou similar	Proteção do ambiente computacional	Média	Médio	O ambiente pode favorecer ou permitir a inserção de código que facilite ou possibilite a veiculação de anúncios indesejados – *adware* – no software
15	Manipulação de mecanismos de busca	Vírus ou similar	Proteção do ambiente computacional	Alta	Baixo	O ambiente pode favorecer ou permitir a inserção de código que simule o acesso a sites específicos com o intuito de incrementar o ranking de acesso a tais sites.
16	Uso indevido de recursos computacionais	Spam	Proteção do ambiente computacional	Alta	Baixo	O ambiente pode favorecer ou permitir a inserção de código que facilite ou possibilite o envio de mensagens de e-mail indesejadas (*spam*).

D.5.1 A análise de risco dos requisitos do software µCRM

Os riscos apresentados até este ponto são decorrentes de ameaças típicas dos ambientes de computação móveis e podem ser considerados genéricos, quer seja, são ofensores a todo e qualquer tipo de software característicos destes ambientes. A seguir são avaliados de forma mais específica os riscos do software µCRM decorrente dos requisitos anotados em sua especificação.

É importante ressaltar que os riscos decorrentes dos requisitos podem derivar-se ou sobrepor-se aos riscos genéricos, agravando ou reduzindo os impactos desses ou amplificando as vulnerabilidades.

Quadro 8 - Riscos dos requisitos.

Item	Requisito	Ameaça	Vulnerabilidade	Probabilidade	Impacto	Comentário
17	Monitorar o uso das funções de telefone	*Malware*	Comunicação inter-processos do Android	Alta	Muito Alto	Um atacante pode contaminar a aplicação e acessar os dados uso do telefone sem conhecimento do usuário.
18	Monitorar o uso das mensagens de texto	*Malware*	Comunicação inter-processos do Android	Alta	Alto	Um atacante pode contaminar a aplicação e acessar os dados uso de mensagens sem conhecimento do usuário.
19	Permitir a associação de um conteúdo textual ao registro de ligações e mensagens	Falha	Disponibilidade dos serviços de acesso ao banco de dados	Baixa	Médio	O serviço de acesso ao banco de dados pode ficar comprometido ou inoperante, comprometendo a integridade devido à impossibilidade de uma atualização.
20	Armazenar conteúdo associado em um repositório específico	Falha	Disponibilidade dos serviços de acesso ao banco de dados	Baixa	Médio	O serviço de acesso ao banco de dados pode ficar comprometido ou inoperante, comprometendo a integridade devido à impossibilidade de uma atualização.
21	Acessar conteúdo associado em um repositório específico	Falha	Disponibilidade dos serviços de acesso ao banco de dados	Baixa	Médio	O serviço de acesso ao banco de dados pode ficar comprometido ou inoperante, comprometendo a disponibilidade.

D.5.2 A classificação e priorização no tratamento dos riscos

Uma vez identificados os riscos aos quais está sujeito o software, cabe estabelecer uma classificação dos riscos quanto à prioridade no tratamento. Para o estabelecimento destas prioridades pode-se utilizar uma tabela simples com os níveis de probabilidade X impacto, na qual são inseridas estas características de cada um dos riscos apontados, apresentando assim de uma maneira rápida o panorama dos riscos e a necessária providência de contramedidas. Um exemplo deste estabelecimento pode ser o apresentado na Tabela 11, que enumera os riscos em cada célula considerando os níveis citados.

Tabela 11 - Classificação e priorização do tratamento dos riscos.

PROBABILIDADE						
	5					
	4	#15, #16			#1, #3, #5, #18	#17
	3	#6		#8, #14	#13	
	2			#9	#2, #4, #7, #10, #11, #12	
	1			#19, #20, #21		
Legenda: Alto, Médio, Baixo		1	2	3	4	5
		IMPACTO				

D.5.3 As contramedidas

Elencados os riscos e estabelecidas as prioridades de tratamento, cabe a avaliação de contramedidas, considerando-se o tratamento dos riscos como abordado na Seção 4.5. É essencial considerar o tratamento dos riscos em função da prioridade, iniciando-se pelos itens classificados mais acima e à esquerda – área mais escura – da tabela de classificação. Outra medida importante para a definição de contramedidas é a consideração dos custos perante os resultados possíveis, o que pode ser obtido com uma análise quantitativa dos riscos e do cálculo do ROI - Retorno do Investimento destas contramedidas.

Como proposta de contramedidas para os riscos apresentados tem-se, por exemplo:

- Para o item #3, um monitoramento do nível de carga da bateria antecedendo a realização de operações críticas – como a gravação ou a transferência de dados em grande volume;
- Para o item #12, o uso de criptografia e *tokens* durante a comunicação por meio de *Bluetooth*;

D.6 As máquinas de ataque

Partindo-se dos casos de uso impróprio elencados é possível modelar as máquinas de ataque que possam reproduzir ou simular a ocorrência descrita, efetivando-se assim o risco, com o intuito de comprovar a eficácia das contramedidas adotadas. Os modelos de máquina de ataque propostos apresentam as informações obtidas desde o início do processo de desenvolvimento do software, como já exposto. Estas informações são avaliadas e consolidadas em um documento único, que contém:

- A **identificação** do modelo da máquina de ataque, isto é, um identificador único para o modelo durante todo o processo, que permite a rastreabilidade do documento e a sua vinculação com os demais processos estabelecidos;
- A **versão** do documento;
- A **descrição** da máquina de ataque ou da classe de máquinas de ataque com o objetivo ou características do ataque por ela modelado;
- O registro do **autor** e da **data** do documento;
- A identificação do **projeto** para o qual a máquina de ataque se destina;
- O **diagrama UML** com a representação gráfica das classes que compõem a máquina de ataque em questão, sua hierarquia e suas interações;
- O **algoritmo** utilizado para o projeto da máquina de estado, o qual fundamentará a construção do código de sua implementação;
- Uma análise com base no modelo **AIDD – *Attack Identification Description and Defense,*** proposto por Soussi (2014). Este modelo visa classificar e caracterizar determinados tipos (classes) de ataques e permite, deste modo, identificar características comuns que podem ser combatidas por defesas específicas. Isto favorece o projetista e o desenvolvedor na construção da máquina de ataque, e também o testador, quando da sua utilização. Além disso, o detalhamento do

ataque perpetrado pela máquina de ataque ora projetada favorece a estruturação das defesas. Essa análise é composta de quatro categorias de informação:

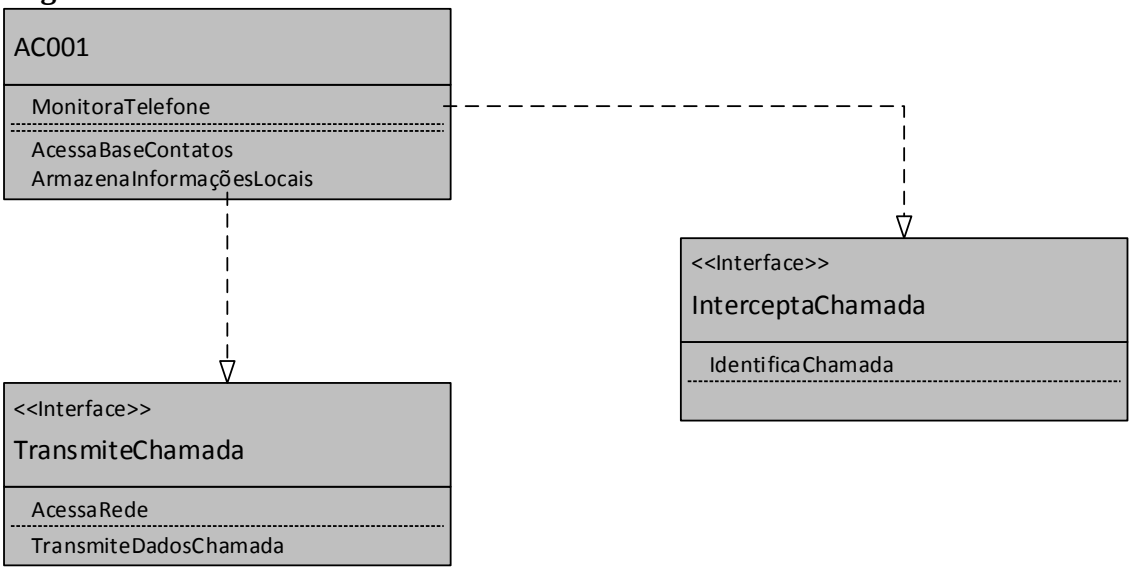
- A **fonte**, isto é, a origem do ataque, que pode ser subdividida em local ou remota, ou seja, o atacante ou ofensor encontra-se no mesmo ambiente computacional do software ou em outro ambiente, acessando este por meio de uma rede;
- O **alvo** do ataque, quer seja, qual componente do ambiente computacional é visado pelo atacante. Este alvo pode ser também uma vulnerabilidade do sistema operacional, uma falha do protocolo de rede partes, funções ou fragilidades da aplicação – do software propriamente dito, que possam ser exploradas com o intuito de possibilitar o ataque.
- O **vetor**, ou seja, o método utilizado pelo atacante para efetivar o ataque. Pode incluir vulnerabilidades, uma vez que estas são necessárias para o sucesso do ataque, incluindo falhas de validação, existência de *exploits*, falha de projeto, *spoofing*, falhas de configuração e engenharia social, entre outros.
- O **resultado** esperado do ataque, do ponto de vista do impacto, que pode incluir uma negação de serviço, a escalação de privilégios, a desativação de defesa, o uso de recursos específicos, a disseminação de código ou a a obtenção de informações, por exemplo. Pode incluir uma falha do ataque, indicando que o mesmo não foi bem-sucedido.

- Os **requisitos** ou regras de negócio que têm relação com a máquina de ataque, quer seja por implicar em uma vulnerabilidade, quer seja por serem afetados pelos ataques ou impactados pelos resultados desse ataque;
- Os **casos de uso impróprio** relacionados com a máquina de ataque em questão, ou seja, aqueles utilizados para a modelagem da mesma.

A seguir são apresentadas as definições de máquinas de ataque elaboradas de acordo com a proposta do SDD.

Id	AM001	**Versão**	1.0
Descrição	Este modelo apresenta as classes da máquina de ataque para a simulação da ocorrência de um ataque no qual o Atacante intercepta o registro de detalhes de uma chamada telefônica realizada para um contato e transmite tais informações para um servidor usando a rede.		
Autor	Luis Gonzaga de Paulo	**Data**	20/03/2015
Projeto	µCRM – Sistema Micro CRM		

Diagrama UML:



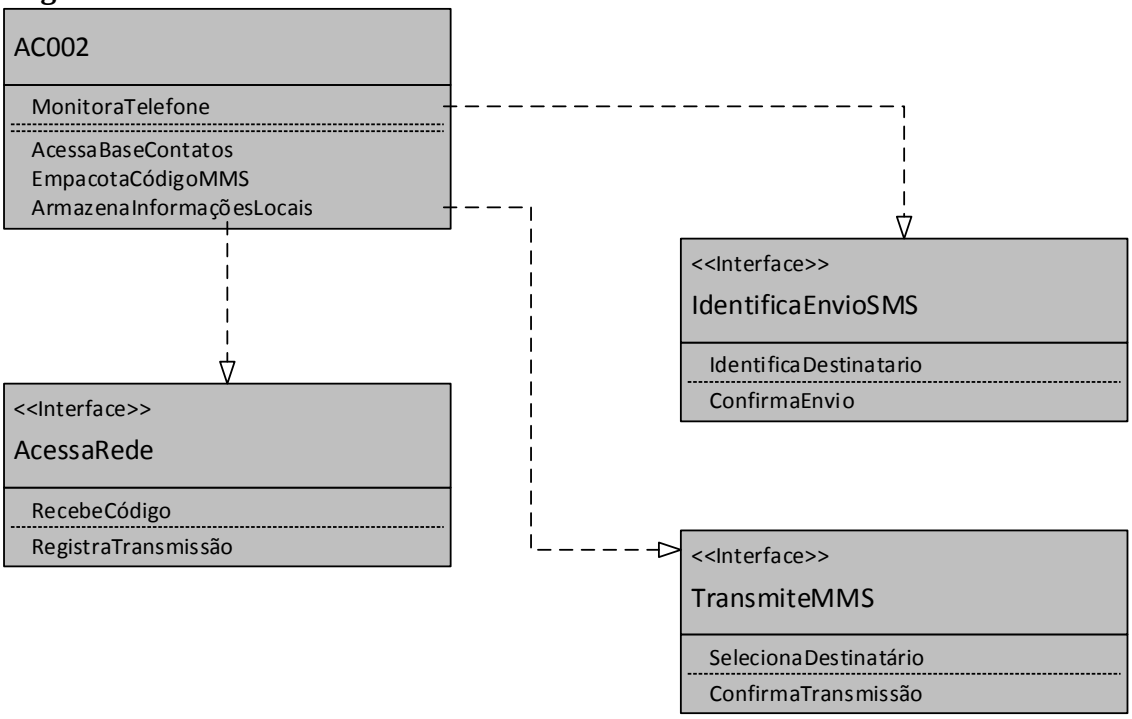
Algoritmo:

```
Faça Enquanto ligado
     Verifica Estado_Telefone
     Se Estado_Telefone = Em_Repouso
     Então verifica Chamada_Realizada
          Se Chamada_Realizada = "OK" E Registra_Chamada Não
          Contém ID_Chamada
          Então
               Captura Dados_Chamada
               Registra_Chamada Armazena ID_Chamada
          Fim-Então
          Se Rede_Disponível = "OK"
          Então
               Acessa_Servidor
               Transmite_Dados
          Fim-Então
          Senão
               Acessa_Base_Local
               Armazena Dados_Chamada
               Registra_Chamada = ""
               Agenda_Evento Transmissao_Dados
          Fim-Senão
     Fim-Então
FIM.
```

AIDD – *Attacks Identification Description and Defense*
Origem: Local – Sistema Operacional ou aplicação instalada no telefone. **Alvo:** Banco de dados da aplicação **Vetor:** Comunicação inter processos e gerenciamento de memória do Android **Resultado:** Exposição dos dados das chamadas realizadas – violação da confidencialidade
Requisitos relacionados: RF2 RF4.3 RF4.4 RF4.5 RF4.6 RNF4
Casos de uso impróprio relacionados: AC001 AC003

Id	AM002	**Versão**	1.0
Descrição	Atacante detecta o envio de SMS para um destinatário da base de dados do software e, por meio da interceptação dos dados, acessa um código malicioso com o intuito de propagá-lo e faz o envio deste código em formato MMS para aquele destinatário.		
Autor	Luis Gonzaga de Paulo	**Data**	20/03/2015
Projeto	µCRM – Sistema Micro CRM		

Diagrama UML:



Algoritmo:

```
Faça Enquanto ligado
     Verifica Estado_Telefone
     Se Estado_Telefone = Envia_SMS
     Então verifica Codigo_Disponivel
          Se Codigo_Disponivel = "OK" E Registra_SMS Não
          Contém ID_SMS
          Então
               Empacota Codigo_MMS
               Registra_SMS Armazena ID_SMS
               Acessa_Rede_Telefone
               Transmite_MMS
          Fim-Então
          Senão
               Acessa_Base_Local
               Armazena Dados_SMS
               Registra_SMS = ""
               Agenda_Evento Transmissao_MMS
          Fim-Senão
     Fim-Então
```

FIM.
AIDD – *Attacks Identification Description and Defense* **Origem:** Local – Sistema Operacional ou aplicação instalada no telefone. **Alvo:** Serviço de envio de mensagens multimídia (MMS) **Vetor:** Comunicação inter-processos e gerenciamento de memória do Android **Resultado:** Envio de conteúdo não desejado. Disseminação de *malware*.
Requisitos relacionados: RF2 RF4.3 RF4.4 RF4.5 RF4.6 RNF4
Casos de uso impróprio relacionados: AC001 AC003

Há três caminhos para o Fracasso:

Não ensinar o que se sabe,

Não praticar o que se ensina,

Não perguntar o que se ignora.

São Beda – Sec. VIII

**** FIM ****