Universidade de Brasília - UnB
Faculdade UnB Gama - FGA
Engenharia de Software

ANÁLISE DE IMPACTO DO USO DE TÉCNICAS DE PROGRAMAÇÃO NO DESEMPENHO DE APLICAÇÕES ANDROID™

Autor: Victor Hugo Alves de Carvalho
Orientador: Msc Fabiana Freitas Mendes

Brasília, DF
2014

Victor Hugo Alves de Carvalho

ANÁLISE DE IMPACTO DO USO DE TÉCNICAS DE PROGRAMAÇÃO NO DESEMPENHO DE APLICAÇÕES ANDROID™

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Msc Fabiana Freitas Mendes

Brasília, DF

2014

Victor Hugo Alves de Carvalho
ANÁLISE DE IMPACTO DO USO DE TÉCNICAS DE PROGRAMAÇÃO NO DESEMPENHO DE APLICAÇÕES ANDROID™/ Victor Hugo Alves de Carvalho. – Brasília, DF, 2014-
52 p. : il. (algumas color.) ; 30 cm.

Orientador: Msc Fabiana Freitas Mendes

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2014.

1. Android. 2. Desempenho. I. Msc Fabiana Freitas Mendes. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. ANÁLISE DE IMPACTO DO USO DE TÉCNICAS DE PROGRAMAÇÃO NO DESEMPENHO DE APLICAÇÕES ANDROID™

CDU 02:141:005.6

Victor Hugo Alves de Carvalho

ANÁLISE DE IMPACTO DO USO DE TÉCNICAS DE PROGRAMAÇÃO NO DESEMPENHO DE APLICAÇÕES ANDROID™

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Trabalho aprovado. Brasília, DF, 25 de junho de 2014:

Msc Fabiana Freitas Mendes
Orientador

Msc Elaine Venson
Convidado 1

Msc Ricardo Ajax Dias Kosloski
Convidado 2

Brasília, DF
2014

Este trabalho é dedicado a minha família e minha namorada que, mesmo nos momentos difíceis não deixaram de me apoiar.

Agradecimentos

À Deus que me deu forças e bençãos para prosseguir minha caminhada.

Aos meus pais e irmãos que nunca me abandoram e sempre acreditaram em mim.

A minha namorada Mírian que contribuiu para o meu desenvolvimento profissional.

Aos meus companheiros de trabalho que deram todo apoio durante os cinco anos que cursei a Universidade.

A minha orientadora, Fabiana Freitas Mendes, pela orientação concedida e compreensão durante a elaboração dos trabalhos.

A todos aqueles de uma forma ou de outra contribuíram para a realização deste trabalho.

Resumo

A plataforma Android cresceu rapidamente desde o seu lançamento, não só em número de dispositivos, mas também na sua presença nos mercados. Por causa da facilidade de se programar, da flexibilidade da plataforma e confiança do mercado, empresas de base tecnológica tem adotado esta plataforma, utilizando sistemas com Android nas mais diversas áreas. Devido à complexidade de se melhorar o desempenho de um aplicativo, este trabalho tem como objetivo comparar o uso e desuso de técnicas de programação por meio de métricas. Para tanto, foi realizada pesquisa aplicada e bibliográfica como metodologia de pesquisa. Foram estudas as técnicas de programação,métricas de software, ferramentas para coleta de dados, e o software para aplicação das técnicas selecionadas. Estes estudos resultaram na implementação de uma técnica de programação que mostrou uma piora no desempenho da aplicação de acordo com a métrica que foi definida.

Palavras-chaves: Android. Desempenho. Software. Métricas de Software. Ferramentas.

Abstract

The Android platform has grown rapidly since its launch, not only in number of devices, but also in its presence in the markets. Because of the ease of programming, the flexibility of the platform and confidence in the market, technology-based companies have adopted this platform using Android systems in several areas. Due to the complexity of improving the performance of an application, this paper aims to compare the use and disuse of programming techniques using metrics. For both applied research and literature was performed as a research methodology. Programming techniques, software metrics, tools for data collection, and software for application of selected techniques were studied. These studies resulted in the implementation of a programming technique that showed a worsening in the application according to the metric that was defined.

Keywords: Android. performance. Software. Software Metrics. Tools.

Lista de ilustrações

Figura 1 – Com Getter e Setter 28
Figura 2 – Sem Getter e Setter 28
Figura 3 – Sem *for* aprimorado 29
Figura 4 – Com *for* aprimorado 29
Figura 5 – Sem uso de ponto flutuante 30
Figura 6 – Com uso de ponto flutuante 30
Figura 7 – Processo de Revisão Sistemática (BRERETON et al., 2006) 31
Figura 8 – String de busca definida 33
Figura 9 – String de busca final definida 33
Figura 10 – GQM Hierarquia (BASILI; CALDIERA; ROMBACH, 1994) 38
Figura 11 – Modelo de Informação de Medição (ISO/IEC, 2007) 39

Lista de tabelas

Tabela 1 – Cronograma TCC 1 22
Tabela 2 – Cronograma TCC 2 23
Tabela 3 – Resumo do que foi feito do Processo de Revisão Sistemática 32
Tabela 4 – Resultado da Pesquisa 34
Tabela 5 – Categoria da informação e conceito mensurável (KOSLOSKI, 2013) . . 40
Tabela 6 – Objetivo de Medição (BERGHOUT; SOLINGEN, 1999) 43
Tabela 7 – Abstraction sheet (BERGHOUT; SOLINGEN, 1999) 44
Tabela 8 – Detalhamento de uma das Métricas 45
Tabela 9 – Sem a implementação da técnica de programação 47
Tabela 10 – Com a implementação da técnica de programação 48

Sumário

1 INTRODUÇÃO . . . 19
1.1 Contexto e Motivação do Estudo . . . 19
1.2 Descrição do Problema . . . 20
1.3 Objetivos Geral e Específico . . . 21
1.4 Metodologia . . . 21
1.5 Cronograma . . . 21

2 TÉCNICAS DE PROGRAMAÇÃO . . . 25
2.1 Escolha do Android . . . 25
2.2 Escolha da Google . . . 25
2.3 Escolha do Foco . . . 26
2.4 Dicas de Desempenho . . . 27
2.4.1 Utilizar estática ao invés de virtual . . . 27
2.4.2 Evite Getters / Setters internos . . . 28
2.4.3 Usar sintaxe aprimorada da estrutura de repetição for . . . 29
2.4.4 Evitar o uso de ponto flutuante . . . 30

3 FERRAMENTAS E SOFTWARE . . . 31
3.1 Protocolo de Pesquisa . . . 31
3.1.1 Objetivo e Questões de Pesquisa . . . 32
3.1.2 Processo e Critérios de Seleção . . . 32
3.1.3 Procedimento para Extração dos Dados . . . 33
3.1.4 Condução da Pesquisa . . . 33
3.2 Ferramentas . . . 34
3.2.1 DDMS . . . 34
3.2.2 Little Eye . . . 35
3.3 Software . . . 36
3.3.1 CPU Spy . . . 36

4 MÉTRICAS DE SOFTWARE . . . 37
4.1 Goal Question Metric . . . 37
4.2 Practical Software Measurement . . . 38
4.2.1 Modelo de Informação . . . 38
4.2.2 Modelo de Processo . . . 39
4.3 Considerações Finais . . . 41

5 MEDIÇÃO . . . 43
5.1 Plano de medição . . . 43
5.1.1 Objetivo de Medição . . . 43
5.1.2 Abstraction sheet . . . 44
5.1.3 Questões e Hipóteses . . . 44
5.1.4 Métricas . . . 45
5.1.5 Detalhamento da Métrica . . . 45
5.2 Coleta . . . 46
5.3 Ambiente de Teste . . . 46
5.4 Resultados . . . 47
5.4.1 Recursos utilizados . . . 47
5.4.2 Resultados obtidos . . . 47

6 CONSIDERAÇÕES FINAIS . . . 49
6.1 Conclusão . . . 49
6.2 Trabalhos Futuros . . . 49

Referências . . . 51

1 INTRODUÇÃO

A necessidade de comunicação em tempo real fez com que o celular alcançasse a utilização em grande escala. Segundo os últimos dados da ANATEL (Agência Nacional de Telecomunicações) (ANATEL, 2013), em 2013 o Brasil fechou com 271,10 milhões de linhas de telefones móvel ativas.

Inicialmente, o celular tinha apenas propósito de prover serviço de telefonia móvel, mas, nos últimos anos ocorreu uma evolução surgindo novos dispositivos que oferecem mais possibilidades de desenvolvimento de aplicações variadas, aumentando, assim, o propósito do celular. O Android é um Sistema Operacional de código aberto que permite acessar qualquer funcionalidade do núcleo do telefone, como chamadas, mensagens, câmera e outras para que sejam adaptadas e aprimoradas visando sempre a evolução do sistema e melhor experiência de uso pelo usuário final (PEREIRA; SILVA, 2009).

Neste documento são abordadas as técnicas de programação, disponibilizadas pelo mantenedor do Android, Google, que afetam o desempenho de um software desenvolvido utilizando a plataforma Android (OHA, 2009). Também será tratada a criação de métricas de qualidade, visando o estabelecimento de dados para comprovação do impacto individual e coletivo das técnicas de programação.

Este capitulo será organizado da seguinte maneira: primeiramente será mostrado o contexto e a motivação de estudo, seguido da descrição do problema, objetivos gerais e específicos, metodologia escolhida, terminando com o cronograma deste trabalho.

1.1 Contexto e Motivação do Estudo

O Android é um sistema operacional baseado em Linux e com código aberto. Foi idealizado para aparelhos com especificações diversas (OHA, 2009) e suporta tecnologias presentes nos smartphones mais modernos atualmente como *touchscreen* (tela sensível ao toque) e GPS (*Global Positioning System* – Sistema de Posicionamento Global) desenvolvido pela Google para smartphones e tablets.

Mesmo tendo sido baseado no sistema Linux, Pereira e Silva (2009) não consideram o sistema operacional Android um Linux completo. Usado e customizado por várias empresas de eletrônicos, o sistema encontra-se em sua versão 4.4, chamada KitKat. Diversas empresas, entre elas a Google, se juntaram para formar, em novembro de 2007, a OHA (Open Handset Alliance) e cujo objetivo é no contexto do Android acelerar e trazer inovação em aplicações e serviços.

A plataforma Android é uma solução completa para tecnologia móvel, oferecendo um pacote com diversos programas para celular, já com um sistema operacional, aplicativos e interface para o usuário.

Uma grande vantagem do sistema é o grande número de desenvolvedores que contribuem na criação de aplicativos. O problema intrínseco a esse processo é o grande número de malwares e aplicativos com falhas de segurança que acabam sendo desenvolvidos em massa. Por ser *open source*, pode ser sempre adaptado a fim de incorporar novas tecnologias. A plataforma está em constante evolução, já que há varias comunidades de desenvolvedores trabalhando em conjunto para construir aplicações móveis inovadoras (PEREIRA; SILVA, 2009).

Pode-se dizer que o desempenho é a relação entre a quantidade de recursos, quantidades de processamento e tempos utilizados para executar certo algoritmo e pode sofrer influência de fatores internos e externos. Fatores internos são instruções provenientes do código fonte, como apresentação de gráficos na tela ou gravação de dados. Fatores externos são variáveis não controladas pelos desenvolvedores do código, como ambiente e rede (PEREIRA; SILVA, 2009).

A busca de desempenho no software para ambientes móveis é complexa, os componentes são variáveis no tempo e espaço em termos de portabilidade, mobilidade e conectividade. Porém, existem também características comuns como a capacidade das aplicações adaptarem sua funcionalidade às condições dos recursos envolvidos nos diferentes momentos da execução (YAMIN et al., 2001).

Este trabalho, portanto, insere-se no contexto de desempenho em aplicativos desenvolvidos utilizando o Sistema Operacional Android.

1.2 Descrição do Problema

Em aplicações para dispositivos móveis o requisito não funcional de desempenho é prioridade. Durante o desenvolvimento de um aplicativo, os recursos utilizados são provenientes da máquina do desenvolvedor e dos ambientes de teste. Ao se distribuir esse aplicativo, ele será utilizado em diferentes ambientes, com baixo ou alto recurso computacional. Por poder ser executado em vários dispositivos e por muitos deles possuírem poucos recursos, o sistema operacional monitora o uso de recursos, podendo até encerrar a execução do aplicativo.

Programadores, em sua formação, aprendem que a aplicação de técnicas de programação, ou seja, boas práticas de desenvolvimento auxiliam a reduzir o esforço de manutenção de software.

São sérias as consequências práticas de desenvolver software de maneira desorde-

nada, sem utilização de processos definidos e de melhores práticas de desenvolvimento de software, por exemplo, softwares difíceis de se dar manutenção, tanto corretiva quanto evolutiva (ENGHOLM, 2010).

Esse trabalho avalia a utilização de técnicas em aplicativos desenvolvidos utilizando o Sistema Operacional Android, buscando mostrar o impactos que essas técnicas tem no desempenho de uma aplicação.

1.3 Objetivos Geral e Específico

Motivado pela necessidade de aplicativos cada vez mais eficientes e pelo crescente número de dispositivos móveis utilizando o sistema operacional Android, o objetivo deste trabalho é avaliar o impacto de um conjunto de técnicas de programação no desempenho de um aplicativo desenvolvido para Android. Para tanto, é necessário atingir os seguintes objetivos específicos:

- Selecionar técnicas de programação e implementa-las em uma aplicação Android, a fim de avaliar se realmente influenciam o desempenho.
- Avaliar o impacto individual de cada técnica de programação no desempenho da aplicação.
- Avaliar o impacto do conjunto das técnicas de programação no desempenho da aplicação.

1.4 Metodologia

De acordo com (SILVA; MENEZES, 2001), a pesquisa utilizada neste TCC pode ser classificada como:

- Pesquisa Aplicada,pois visa produzir conhecimentos para aplicação prática, voltados para o problema de desempenho em aplicações Android.
- Pesquisa Bibliográfica, pois busca informações com o uso de computadores e em informações publicadas em livros, revistas e artigos científico.

1.5 Cronograma

Nas Tabelas 1 e 2, respectivamente, é mostrado o cronograma do TCC 1 e TCC 2. Cada parte está estruturada com uma tabela contendo as etapas e suas respectivas datas de execução, onde cada X corresponte a semana para realização de uma etapa. Essas

etapas serão descritas contendo uma breve explicação seguido do método de pesquisa adotado.

Em relação à TCC 1, foi executado o cronograma apresentado na Tabela 1.

Tabela 1 – Cronograma TCC 1

MESES/SEMANAS																
Etapas	Março			Abril					Maio				Junho			
	1	**2**	**3**	**1**	**2**	**3**	**4**	**5**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**
Etapa 1	x															
Etapa 2		x	x													
Etapa 3				x	x											
Etapa 4					x	x										
Etapa 5						x	x									
Etapa 6								x	x							
Etapa 7										x	x	x	x			
Etapa 8														x		

Neste trabalho, forão utilizados os seguintes métodos de pesquisa:

1. Pesquisa Simples: busca em livros, sites, ferramentas de busca, sem a definição de um método de pesquisa como Revisão Sistemática, Mapeamento Sistemático.

2. Estratégia base no método de pesquisa Revisão Sistemática, porém de uma maneira mais simplificada como pode ser visto no Capítulo 3.

Como pode ser visto na Tabela 1, o trabalho foi dividido em oito etapas, sendo elas:

Etapa 1 – Apresentação do tema.Aqui foi apresentado e discutido tema do TCC para a professora orientadora.
Método de pesquisa: Não Há

Etapa 2 – Inicio do trabalho de conclusão do curso (TCC).Nesta etapa foi elaborada a introdução, contexto e motivação de estudo, descrição do problema, objetivos (geral e específico), cronograma e metodologia.
Método de pesquisa: 1

Etapa 3 – Técnicas de Programação listadas pela Google. Nesta etapa foram selecionadas as técnicas de programação citadas pela Google.
Método de pesquisa: 1

Etapa 4 – Plano de medição. Nesta etapa foram Elaborados os planos de medição.
Método de pesquisa: 1

Etapa 5 – Ferramentas para coleta de dados. Nesta etapa foram selecionadas e analisadas algumas ferramentas que poderiam ser utilizadas para coleta de dados.
Método de pesquisa: 1

Etapa 6 – Software que será analisado. Aqui foram analisadas diversas aplicaçoões Android que poderiam ser utilizadas para testar as técnicas de programação selecionadas.
Método de pesquisa: 1

Etapa 7 - Seleção de técnicas, ferramentas e software. Nesta etapa foram analisadas técnicas, ferramentas e softwares, em seguida, foi selecionada quais as que foram utilizados para a produção do TCC.
Método de pesquisa: 2

Etapa 8 – Revisão do trabalho. Nesta etapa o trabalho escrito deste TCC foi revisado.

Sobre TCC 2, é proposto um cronograma o qual é apresentado na Tabela 2.

Tabela 2 – Cronograma TCC 2

MESES/SEMANAS																
Etapas	**Agosto**				**Setembro**				**Outubro**				**Novembro**			
	1	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**
Etapa 1	x	x	x													
Etapa 2				x	x	x										
Etapa 3							x	x	x							
Etapa 4										x	x	x	x			

Como pode ser visto, a segunda parte de TCC está dividida em quatro etapas, são elas:

Etapa 1 – Desenvolver instrumentos de análise.

Etapa 2 – Configuração do Ambiente.

Etapa 3 – Aplicar os instrumentos.

Etapa 4 – analisar os resultados.

Este documento estará organizado da forma como se segue:

- **Capítulo 2**: Mostra a pesquisa realizada sobre as técnicas de programação que afetam o desempenho de uma aplicação Android, em seguida são apresentadas aquelas que serão alvos de estudo nesse trabalho.
- **Capítulo 3**: Mostra o protocolo de pesquisa definido para a busca de ferramentas que será realizada em TCC 2, as ferramentas e o software que foram utilizados para realizar a coleta dos dados e a implementação da técnica de programação.

- **Capítulo 4**: Mostra a pesquisa realizada sobre as possíveis métricas de software e a escolha de qual será abordada.
- **Capítulo 5**: Define o plano de medição para realizar a coleta dos dados e em seguida mostrado os resultados obtidos.
- **Capítulo 6**: Mostra um resumo do que foi feito neste trabalho e o que será realizado em TCC 2.

2 TÉCNICAS DE PROGRAMAÇÃO

Este capítulo está estruturado da seguinte forma: inicialmente é apresentada a justificativa para a escolha do Android, depois a justificativa para a escolha da Google como referência de estudo, a justificativa da escolha do foco do trabalho, seguido das dicas de desempenho que foram selecionadas.

2.1 Escolha do Android

Com o crescente número de opções de dispositivos Android no mercado, cada vez mais as pessoas se perguntam: "Qual o melhor dispositivo pra mim?" no cenário de dispositivos celulares móveis, existem muitas formas de se obter informações que respondam a essa pergunta. Vendedores em lojas físicas, comentários em lojas virtuais e recomendações de amigos, colegas e afins, são as primeiras opções de leigos.

Já para aqueles que têm conhecimento aprofundado das tecnologias que compõem um celular, essas recomendações não são sempre suficientes. Para uma análise mais aprofundada, é preferível recorrer a análises de desempenho ou comparativos entre dispositivos.

Por ser um sistema operacional que vem crescendo com o decorrer do tempo e por ter programas desenvolvidos seguindo o princípio de software livre, ou seja, programa que pode ser copiado, usado, modificado e redistribuído de acordo com as necessidades de cada usuário (FSF, 2004); foi escolhido o Android como sistema operacional que será utilizado neste TCC.

2.2 Escolha da Google

Mantido pela Google, o sistema operacional é aberto, baseado em Linux, dividido em camadas e compatível com diversos dispositivos. A flexibilidade e abertura do sistema fez com que milhares de pessoas contribuíssem para o desenvolvimento da tecnologia Android.

Como dito acima, a Google é responsável por manter o sistema operacional Android, com isso ela desenvolveu um treinamento para desenvolvedores que queiram desenvolver aplicativos para a plataforma.

Por ser a principal empresa responsável pela criação e manutenção no sistema operacional Android, ela foi escolhida como fonte para retirar as técnicas que serão aplicadas no software determinado para obter resultados.

2.3 Escolha do Foco

A Google disponibiliza, em seu site um treinamento para desenvolvedores em Android (GOOGLE, 2013), aulas que descrevem como realizar uma tarefa específica e mostra exemplos de códigos que podem ser utilizados como base para desenvolver um aplicativo.

As aulas estão divididas nos grupos apresentados a seguir:

- **Aula 1** Introdução: ensina o essencial para o desenvolvimento Android, exemplos de atividades, como construir um aplicativo, construção de uma interface de usuários dinâmica e como interagir com outros aplicativos.

- **Aula 2** Aplicativos de construção com compartilhamento de conteúdo: ensina a criar aplicativos que podem compartilhar dados entre dispositivos e aplicativos.

- **Aula 3** Criação de aplicativos com multimídia: ensina a criar aplicações ricas em multimídia que se comportam de acordo com a expectativa do usuário. Exemplos de atividades: gerência de reprodução de áudio e captura de fotos.

- **Aula 4** Construção de aplicativos com gráficos e animação: ensina a construção de uma interface mais robusta, com a realização de algumas tarefas que podem fazer a diferença na concorrência com outros aplicativos. Exemplos de atividades: exibir mapas de bits de forma eficiente e adicionar animações.

- **Aula 5** Construção de aplicativos com conectividade em nuvem: ensina como conectar a outros dispositivos, conectar-se a internet, backup e sincronização de dados, por exemplo, conexão de dispositivos sem fio, executar operações de rede, sincronizar para a nuvem, resolvendo conflitos salvos na nuvem.

- **Aula 6** Construção de aplicativos com informações de usuário e localização: ensina como adicionar a personalização do usuário ao aplicativo. Exemplos de atividades: acessar dados conectados.

- **Aula 7** Melhores práticas para interação e engajamento: ensina a manter os usuários através de implementação de padrões de interação em Android. Exemplos de atividades: planejar e implementar navegação eficaz, notificar usuário, adicionar funcionalidade de pesquisa.

- **Aula 8** Melhores práticas para interface de usuário: ensina construir uma interface usando layouts Android para todos os tipo de aplicativos. Exemplos de atividades: projetar para múltiplas telas, projetar para televisão, implementar acessibilidade e criar interfaces personalizadas.

- **Aula 9** Melhores práticas para a entrada do usuário: ensina a gerenciar *touch screen* e entrada de textos. Exemplos de atividades: usar gestos de toque, manuseio de entrada de teclado, apoiar controladores de jogo.
- **Aula 10** Melhores práticas para serviços de fundo: ensina a executar procedimentos para aumentar o desempenho do aplicativo e diminuir o consumo da bateria. Exemplos de atividades: executando em um serviço de fundo e carregando dados em segundo plano.
- **Aula 11** Melhores práticas para obter aumento no desempenho: ensina a desenvolver um aplicativo mais leve, rápido, menor consumo possível de memória e de bateria. Exemplos de atividades: gerenciar a memória do aplicativo, dicas de desempenho, melhorar o desempenho do layout, otimizar a vida da bateria.
- **Aula 12** Melhores práticas para a segurança e privacidade: ensina como manter seguro os dados do aplicativo. Exemplos de atividades: dicas de segurança, segurança com HTTPS e SSL.
- **Aula 13** Melhores práticas para teste: ensina maneiras eficientes de testar o aplicativos. Exemplo de atividade: teste a aplicação.

Este trabalho focará em analisar as melhores práticas para o desempenho, focando nas técnicas de programação que podem afetar o desempenho de uma aplicação Android, ou seja, concentrará no conteúdo referente a Aula 11, em especial, nas atividade relacionada a dicas de desempenho.

Foi escolhido esse foco, pois o desenvolvimento para dispositivos móveis tende a possuir restrições. Com isso a aplicação das técnicas de programação pode ser o diferencial no desempenho de uma aplicação Android. Além disso, é a única aula que contém dados numéricos que podem ser utilizados como parâmetros de análise deste trabalho.

2.4 Dicas de Desempenho

Nesta etapa serão detalhadas as quatro técnicas de programação que foram selecionadas através do critério de seleção: técnicas que apresentam dados numéricos de ganho de desempenhos ao serem utilizadas. Com isso, esses dados numéricos apresentados serão utilizados como referência para verificar se, de fato, o uso das técnicas tem o benefício afirmado pela Google.

2.4.1 Utilizar estática ao invés de virtual

Esta técnica afirma que, caso seja preciso obter o valor de campos de um objeto, deve-se fazer o uso do método estático. Essa forma de trabalhar é considerada uma boa

prática de programação, pois a assinatura do método que fizer a chamada a este método estático não poderá alterar o estado do objeto.

De acordo com a Google, essa boa pratica afeta o desempenho de uma aplicação, fazendo com que ela fique de 15% a 20% mais rápida.

2.4.2 Evite Getters / Setters internos

A utilização de getters e setters é muito comum nas linguagens orientadas a objetos, pois implementa os conceitos de encapsulamento. Devido a isso, é quase que obrigatória a utilização de métodos para atribuir ou recuperar o valor do atributo de um objeto. Porém essa é uma ideia ruim para Android, pois métodos são mais custosos que acessos diretos a atributos de objetos.

De acordo com a Google, a utilização de acessos diretos aos atributos pode ser de três até sete vezes mais rápido que a utilização de métodos para se obter o valor do atributo. Portanto, a utilização dessa técnica pode ter uma alteração considerável no desempenho da aplicação, e por isso foi selecionada para este projeto de pesquisa.

A Figura 1 apresenta um trecho de código com Getter e Setter.

Figura 1 – Com Getter e Setter

```
private int atributo;

public int getAtributo() {
    return atributo;
}

public void setAtributo(int atributo) {
    this.atributo = atributo;
}
```

```
classe.setAtributo(1);
int atributo = classe.getAtributo();
```

O mesmo trecho de código é apresentado na Figura 2 sem Getter e Setter.

Figura 2 – Sem Getter e Setter

```
public int atributo;
```

```
classe.atributo = 1;
int atributo = classe.atributo;
```

2.4.3 Usar sintaxe aprimorada da estrutura de repetição for

A estrutura de repetição *for* é muito utilizada ao se programar e tem a varredura de listas como uma de suas principais utilizações. Na maioria das linguagens de programação, a maneira como a lista é percorrida não influencia o desempenho de maneira significante, porém para Android esse fato não é verdadeiro, e a utilização de um *for* aprimorado, também conhecido como *for each*, pode alterar o desempenho de uma aplicação.

O *for* aprimorado pode ser utilizado para percorrer coleções que implementam a interface Iterable. Sua utilização, de acordo com a Google, pode ser três vezes mais rápida que a utilização de um índice para controlar os acessos à lista.

A Figura 3 apresenta um exemplo de código sem uso do *for* aprimorado.

Figura 3 – Sem *for* aprimorado

```
static class classe() {

    int atributo;
}

Classe[] lista = new Classe[1];

public void funcao(){
    int soma = 0;
    for (int i = 0; i< lista.length; i++){
        soma += lista[i].atributo;
    }

}
```

A Figura 4 apresenta um exemplo de código com uso do *for* aprimorado.

Figura 4 – Com *for* aprimorado

```
static class classe() {

    int atributo;
}

Classe[] lista = new Classe[1];

public void funcao(){
    int soma = 0;
    for (Classe classe : lista){
        soma += classe.atributo;
    }

}
```

A grande diferença entre os codigos das Figuras 3 e 4, é que o uso do *for* aprimorado faz com que a utlização do índice como controle de acesso à lista seja descartado.

2.4.4 Evitar o uso de ponto flutuante

De acordo com a Google, o uso de ponto flutuante, em aplicativos Android, é cerca de duas vezes mais lento do que o uso número inteiro.

A Figura 5 apresenta um exemplo de código sem uso de ponto flutuante.

Figura 5 – Sem uso de ponto flutuante

```
public int valor = 1;
```

A Figura 6 apresenta um exemplo de código com uso de ponto flutuante:

Figura 6 – Com uso de ponto flutuante

```
public float valor = 2.00f;
```

3 FERRAMENTAS E SOFTWARE

Este capítulo mostrará o protocolo de pesquisa definido para buscar ferramentas para a coleta de dados em TCC 2, as ferramentas que serão utilizadas para coletar dados em TCC 1, além de definir o aplicativo escolhido para ser utilizado na aplicação das técnicas de programação definidas no Capítulo 2.

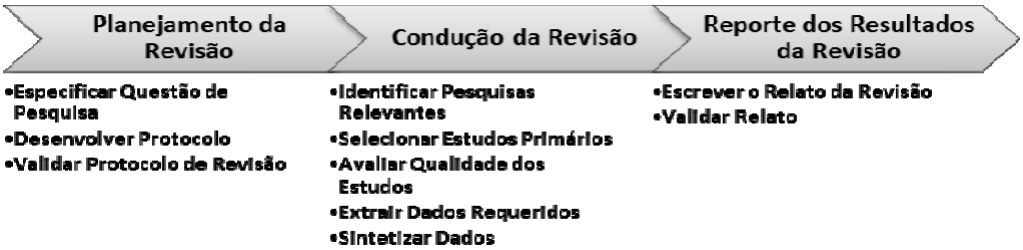
3.1 Protocolo de Pesquisa

Foi utilizada a revisão sistemática como inspiração para definir o método de pesquisa para encontrar ferramentas para coleta de dados sobre desempenho em aplicativos desenvolvidos utilizando o sistema operacional Android.

A técnica de Revisão sistemática consiste em um explícito e rigoroso método para identificar, criticar e sintetizar estudos relevantes de um tópico em particular (BRERETON et al., 2006).

Os procedimentos gerais para realizar uma revisão sistemática em engenharia de software são: planejamento da revisão, em que ocorre a identificação das necessidades para a revisão e elaboração do protocolo; condução da revisão, em que ocorrem a identificação da pesquisa, a seleção e avaliação da qualidade dos estudos, e a extração e síntese dos dados; e reporte dos resultados da revisão, que permite que leitores possam criticar e replicar a revisão (BRERETON et al., 2006). A Figura 7 ilustra este processo.

Figura 7 – Processo de Revisão Sistemática (BRERETON et al., 2006)



Como dito, o processo de Revisão Sistemática, definido na Figura 7, não foi executado em sua totalidade. A Tabela 3 mostra o que foi Realizado, o que foi Realizado Parcialmente e o que Não foi Realizado.

Tabela 3 – Resumo do que foi feito do Processo de Revisão Sistemática

Realizado	**Realizado Parcialmente**	**Não foi Realizado**
Especificar Questão de Pesquisa	Escrever o Relato da Revisão	Validar Protocolo de Revisão
Desenvolver Protocolo	Extrair Dados Requeridos	Avaliar Qualidade dos Estudos
Identificar Pesquisas Relevantes		Sintetizar Dados
Selecionar Estudos Primários		Validar Relato

3.1.1 Objetivo e Questões de Pesquisa

O objetivo desta pesquisa é analisar relatos de ferramentas utilizadas para coletar dados referentes ao desempenho em aplicações Android com propósito de selecionar ferramentas que podem ajudar a coletar dados referentes ao desempenho no contexto de desenvolvimento Android.

A partir do objetivo estabelecido anteriormente, foi elaborada a seguinte questão de pesquisa:

- Q1. Quais são as ferramentas mais utilizadas para coleta de dados referentes a desempenho em aplicações desenvolvidas utilizando o sistema operacional Android?

3.1.2 Processo e Critérios de Seleção

A estratégia adotada é composta por três etapas:

- Identificação das fontes: Consiste na seleção das fontes para se realizar as pesquisas manuais e automáticas.
- Definição da string de pesquisa: Realizada de maneira objetiva.
- Condução da pesquisa automatizada: Utiliza as strings para realizar as pesquisas na fonte.

Para aplicações das strings de busca foi selecionada dentre as bibliotecas digitais o *IEEEXplore* para a busca dos artigos, por ser uma máquina de busca específica para Engenharia de Software.

Os critérios para definir se resultados encontrados serão selecionados são:

1. Os artigos devem utilizar a língua inglesa.
2. Os artigos disponíveis para download ou leitura de forma gratuita.
3. Os artigos devem apresentar estudo sobre ferramentas ou qualquer outro item relacionados ao desempenho em aplicações Android.

Os critérios para definir se resultados encontrados serão excluídos são:

1. O artigo trate o tema de maneira superficial.
2. O Artigo utilize ferramentas para a coleta de dados sobre desempenho em aplicações Android.

3.1.3 Procedimento para Extração dos Dados

Como estratégia de extração de informações, considerou-se que para cada artigo selecionado, serão extraídas as seguintes informações: título, autor(es), data de publicação e a informação da ferramenta.

3.1.4 Condução da Pesquisa

Para realizar a pesquisa foi definida a string de busca criada a partir do objetivo desta pesquisa bibliográfica.

Figura 8 – String de busca definida

```
("android" AND "performance")
```

Muitos artigos forma retornados pela execução da string de busca da Figura 8, por isso foram feitas algumas mudanças na string. O resultado final pode ser observado na Figura 9.

Figura 9 – String de busca final definida

```
(Android AND performance AND (tool OR application OR app OR software) AND measurement)
```

O resultado da execução da string de busca:

Tabela 4 – Resultado da Pesquisa

STRING DE BUSCA	FONDE DE BUSCA	RESULTADOS
Inicial	IEEEXplore	419
Final		32

3.2 Ferramentas

Esta seção lista ferramentas utilizadas neste trabalho juntamente com suas respectivas dependências.

Outras ferramentas poderam ser incluidas, no TCC 2, de acordo com o retorno da pesquisa que foi definida no Seção 3.1.

3.2.1 DDMS

Os dispositivos Android vêm nativamente com uma ferramenta de depuração chamada Dalvik Debug Monitor Server (DDMS), ou Servidor de Monitoramento de Debug do Dalvik, que fornece serviço de captura de tela, informações de Heap e Threads do dispositivo, estado do LogCat, processos e intensidade do sinal recebido pela antena do dispositivo, e ainda coleta informações de chamadas, mensagens de texto e dados de localização (DEVELOPER, 2013).

No Android, cada aplicativo é executado em seu próprio processo, cada um dos quais é executado em sua própria Máquina Virtual (VM). Cada VM expõe uma única porta que um depurador pode anexar. Quando DDMS é iniciado, ele se conecta ao Android Debug Bridge (ADB). Quando um dispositivo for conectado ao computador, um serviço de monitoramento de VM é criada entre ADB e o DDMS, que notifica o DDMS quando uma VM no dispositivo é iniciado ou encerrado. Uma vez que a VM está rodando, o DDMS recupera ID do processo da VM (PID), via ABD, e abre uma conexão com depurador do VM, através do daemon do ABD (ADBD) do dispositivo. DDMS pode agora falar com a VM usando um protocolo personalizado (DEVELOPER, 2013).

Abaixo são mostradas as ferramentas que deverão ser instaladas para sua utilização:

- Software Development Kit (SDK) do Android: Fornece aos desenvolvedores todas as ferramentas para desenvolver um aplicativo para Android, desde o ambiente de desenvolvimento, até ferramentas para gerenciar o download e instalação de novos pacotes e configurações para as versões mais atuais do Android (DEVELOPER, 2013).

- Android Developer Tools (ADT), ou Ferramentas para o Desenvolvedor Android. É um plugin para o Eclipse (ECLIPSE, 2013) que integra nesta IDE todas as ferramentas do SDK. Ele oferece vários recursos que ajudam a desenvolver aplicações para Android rapidamente e com segurança. Além da integração com o Eclipse, que é uma IDE bem estabelecida e conhecida pelos desenvolvedores, o ADT oferece ferramentas visuais para diversos comandos do SDK, que geralmente seriam acessados somente através de linhas de comando

- Android Debug Bridge (ADB) é uma ferramenta de linha de comando versátil que permite a comunicação com uma instância do emulador ou dispositivo Android conectado ao computador via USB.

- O sistema de Log Android fornece um mecanismo para coleta e visualização de saída de depuração do sistema, registros de várias aplicações e partes do sistema são recolhidos numa série de buffers circulares, que podem então ser visualizados pelo comando *logcat* (DEVELOPER, 2013).

3.2.2 Little Eye

Little Eye é uma ferramenta de análise de desempenho e monitoramento que pode ajudar a identificar e corrigir problemas de desempenho em uma aplicação Android com versões a partir da 2.3 (RANGARAJAN et al., 2013).

Esta ferramenta suporta a coleta de métricas sobre CPU, recursos de rede, memória RAM, consumo de energia, GPS e espaço em disco. Tem como principais características (RANGARAJAN et al., 2013):

- **Medir:** mede o desempenho e consumo de recursos do aplicativo. Obtém estatísticas detalhadas sobre a quantidade de energia, dados, CPU, memória e espaço em disco que o aplicativo está utilizando.

- **Analisar:** informa detalhadamente o contexto em torno do aplicativo e os dados coletados, possibilitando uma compreensão muito mais profunda do comportamento do aplicativo.

- **Otimizar:** obtém rapidamente acionáveis "próximos passos" para otimizar o consumo de recursos do aplicativo, seja problemas de memória, bateria, uso excessivo de dados ou até mesmo falhas frequentes.

Para a instalação desta ferramenta, é necessário (RANGARAJAN et al., 2013):

1. Java JRE ou SDK - V 1.6 + (Java 6 ou superior).
2. SDK Android.
3. Depuração USB está ativada no telefone.
4. Se você estiver usando o Windows, pode ser necessário para garantir que você está se conectando como um "Camera (PTP)" e não como um "dispositivo de mídia (MTP)".
5. Se você estiver usando o Windows, você pode precisar instalar os drivers do seu telefone.

3.3 Software

Foi definido que será utilizado software livre, ou seja, programa que pode ser copiado, usado, modificado e redistribuído de acordo com as necessidades de cada usuário (FSF, 2004).

Para procurar esses possíveis softwares foi utilizado o *Git Hub* (HUB, 2014), um site no qual é possivel hospedar softwares, podendo ser compartilhado com outros usúarios.

3.3.1 CPU Spy

Aplicativos simples para exibir o tempo que o CPU gasta em cada estado de frequência. Esta pode ser uma ferramenta útil no diagnóstico de problemas de bateria ou aprimorando suas configurações, também exibe as informações do kernel atual.

Para ter acesso ao código fonte deste aplicativo, basta baixar o artigo disponibilizado no endereço:<https://github.com/bvalosek/cpuspy>.

Este aplicativo foi escolhido pois possibilita a apliacação de todas as técnicas de programação que foram selecionadas no Capítulo 2.

4 MÉTRICAS DE SOFTWARE

De forma geral, pode-se dizer que medição é o processo pelo qual símbolos ou números são associados a atributos do mundo real de forma a descreve-las de acordo com as regras definidas (BERGHOUT; SOLINGEN, 1999). Segundo esses mesmos autores, medição em software é um processo contínuo de definir, coletar e analisar dados com respeito ao processo de software e os seus produtos. A principal razão para o uso de medições é o fato de esperar que produzam um entendimento sobre o processo de software que o torne controlável, além disso, fornecer informações que permitem aperfeiçoá-lo.

Neste capítulo serão estudadas as abordagens GQM – Goal Question Metrics – e PSM – Practical Software Measurement – utilizadas comumente para definição de métricas. Em seguida, é apresentada a abordagem escolhida juntamente com os argumentos que levaram a escolha.

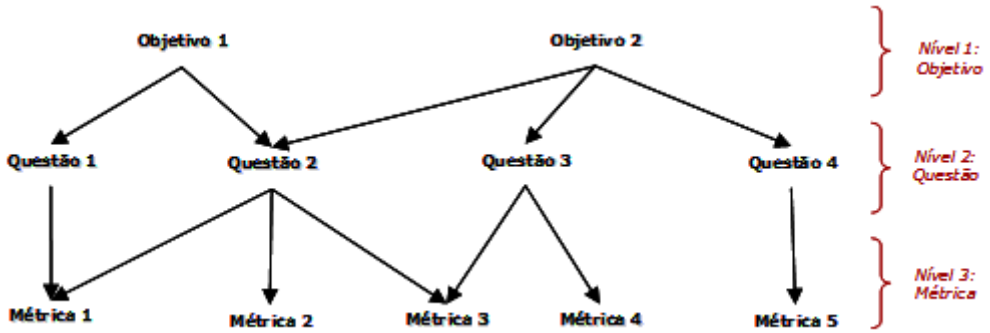
4.1 Goal Question Metric

GQM – *Goal Question Metric* – abordagem definida por Victor Basili (BASILI; CALDIERA; ROMBACH, 1994) é utilizada como guia na definição de métricas para um processo de mensuração. Ela foi originalmente desenvolvida para avaliações de defeitos de um conjunto de projetos no ambiente da NASA, NASA *Goddard Space Flight*, no início da década de 80.

Para o uso do GQM, a abordagem mais indicada para a criação de métricas no âmbito organizacional deve proceder da forma *top-down*, onde inicialmente são encontrados os objetivos que precisam ser respondidos. Diante desses objetivos definidos, um conjunto de questões é definida e por fim são criadas as medições que satisfaçam os objetivos. Vários modelos GQM também podem possuir questões e métricas em comum, mas é importante lembrar que os pontos de vista podem mudar, portanto, devem ser levados em consideração nas medições (BASILI; CALDIERA; ROMBACH, 1994).

A Figura 10 ilustra a hierarquia de objetivos, questões e métricas apresentadas anteriormente.

Figura 10 – GQM Hierarquia (BASILI; CALDIERA; ROMBACH, 1994)



4.2 Practical Software Measurement

PSM – *Practical Software Measurement* – é um modelo para mensuração de projetos de software criado em 1994, sob o patrocínio do Departamento de Defesa Norte Americano. Em 1997 foi determinado um projeto para construção de uma ISO na qual o PSM foi utilizado como base para a criação da norma ISO/IEC 15939 (ISO/IEC, 2007).

O processo de medição proposto por essa ISO é orientado às necessidades de informação da organização. Para cada necessidade de informação, o processo gera um produto de informação para sanar a necessidade.

O PSM é um modelo que estrutura medições em atividades de projetos de software. Nesse sentido, objetiva medir e melhorar processos, projetos e produtos de software. Em nível prático o PSM aborda dois problemas:

- Como especificar de uma maneira formal as medidas que deverão ser utilizadas no projeto, ou seja, especificar formalmente os indicadores a serem usados.
- Como deverá ser conduzido o processo de medição.

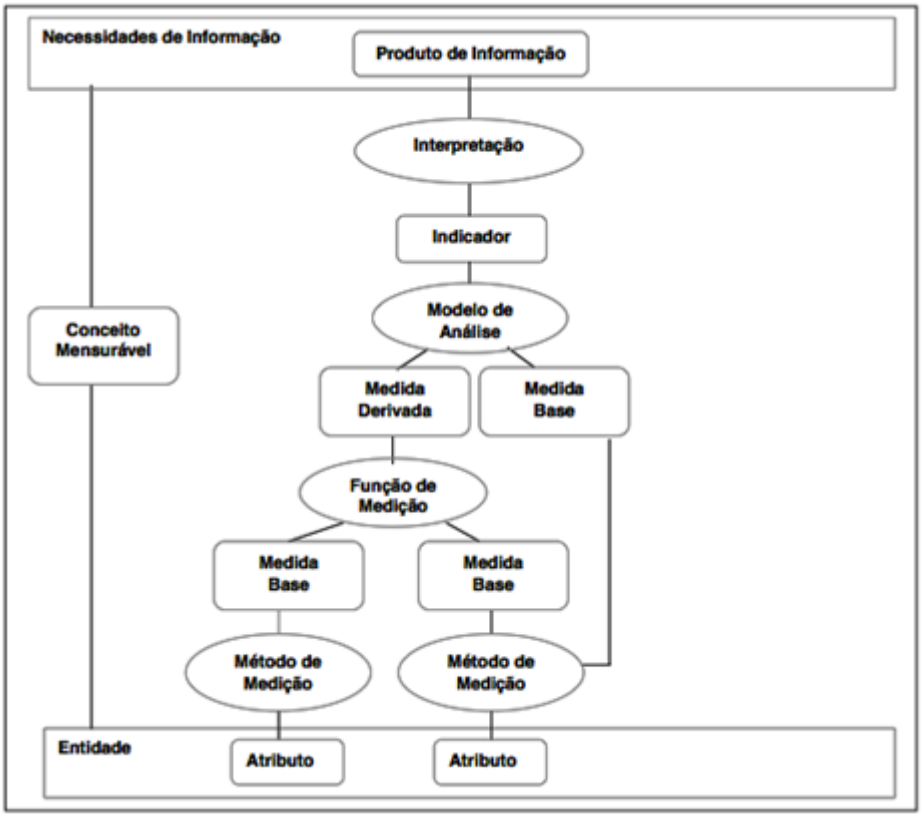
Para atingir tais objetivos, o PSM faz uso do Modelo de Informação e o modelo de processo, os quais são detalhados nas subseções seguintes.

4.2.1 Modelo de Informação

O Modelo de Informação do PSM é uma estrutura para a definição das medidas que deverão ser utilizadas no projeto (ver Figura 11). Para isso o PSM trabalha com o conceito de necessidades de informações mensuráveis que, por sua vez, resultam de esforços dos gerentes para influenciar as saídas de projetos, processos e iniciativas oriundas de objetivos

de medição. As necessidades de informações são usualmente derivadas de duas fontes: objetivos que os gerentes procuram atingir em seus projetos e obstáculos que impedem de atingir tais objetivos (MCGARRY et al., 2001).

Figura 11 – Modelo de Informação de Medição (ISO/IEC, 2007)



4.2.2 Modelo de Processo

Serve de guia a implementação do PSM. É o processo para o estabelecimento, planejamento, execução e avaliação da medição do projeto, organização ou empreendimento como um todo (ISO/IEC, 2007).

A seguir são apresentadas as cinco perspectivas que são centrais para o processo de medição:

1. **Desempenho**: refere-se aos valores característicos que são observados quando medidos os atributos dos produtos e serviços, advindos do processo. Pode ser descrito de duas formas: ao se medir os atributos que os processos produzem e ao se medir os atributos do processo em si.

2. **Estabilidade**: refere-se à habilidade de uma organização em produzir produtos de acordo com um plano e em melhorar processos visando produtos melhores e mais competitivos.

3. **Conformidade**: refere-se ao processo estar claramente definido, efetivamente suportado, fielmente executado e reforçado.

4. **Capacidade**: refere-se aos valores e características que irão variar conforme o passar do tempo. Quando um processo é estável, este possuirá meios previsíveis dentro de intervalos previsíveis. Então é possível examinar se o processo está ou não atingindo os objetivos.

5. **Melhora e Investimento**: refere-se a objetivos de negócio e estratégias, os quais juntamente com dados factuais sobre os atributos de qualidade do produto e desempenho do processo, são as chaves que levam a ações que aperfeiçoam o processo de software.

A Tabela 5 mostra as sete categorias de informação definidas pelo PSM com seus respectivos conceitos mensuráveis que são uma ideia sobre as entidades que poderiam ser medidas para satisfazer uma necessidade identificada.

Tabela 5 – Categoria da informação e conceito mensurável (KOSLOSKI, 2013)

CATEGORIA DA INFORMAÇÃO	CONCEITO MENSURÁVEL
Cronograma e Progresso	Alcance dos Marcos do Projeto
	Desempenho do Caminho Crítico
	Progresso da Unidade de Trabalho
	Capacidade Incremental
Recurso e Custo	Esforço da Equipe
	Desempenho Financeiro
	Recursos Ambientais e de Suporte
Tamanho e Estabilidade do Produto	Tamanho e Estabilidade Físicos
	Tamanho e Estabilidade Funcionais
Qualidade do Processo	Corretude Funcional
	Facilidade de Manutenção
	Eficiência
	Portablidade
	Usabilidade
	Confiabilidade
Desempenho do Processo	Flexibilidade do Processo
	Eficiência do Processo
	Efetividade do Processo
Efetividade da Tecnologia	Adequabilidade da Tecnologia
	Volatilidade da Tecnologia
Satisfação do Cliente	Feedback do Cliente
	Suporte ao Cliente

O PSM aborda o chamado construtor de medição, que é uma estrutura que relaciona atributos específicos de processos e/ou produtos, que podem ser medidos com uma informação específica necessária. O construtor de medição define três níveis de medida:

medida básica, medida derivada e indicador. Cada um desses níveis de medida acrescenta regras específicas para determinar valores e definir métodos, funções e modelos para a medição (MCGARRY et al., 2001).

4.3 Considerações Finais

Para este trabalho será adotada o *Goal Question Metric* (GQM), pois as medições serão orientadas a objetivos e por ter uma metodologia simples e eficaz que atende melhor os objetivos deste trabalho.

5 MEDIÇÃO

Após a escolha das ferramentas para coleta e do software, no Capítulo 3; da abordagem GQM, no Capítulo 4; será estabelecido o plano de medição e em seguida, será definido o procedimento para a coleta dos dados e o ambiente de teste que utilizado. Por fim serão mostrados os resultados obtidos com a execução do plano.

5.1 Plano de medição

Esta seção detalha o plano de medição seguindo a estrutura definida: objetivo de Medição, Seção 5.1.1; abstraction Sheet, Seção 5.1.2; questões e hipóteses, Seção 5.1.3; métricas, Seção 5.1.4; detalhamento da métrica , Seção 5.1.5; resultados, Seção 5.4.2

5.1.1 Objetivo de Medição

Metas devem existir, devem ser identificadas e devem ser o foco para o estabelecimento de metas para medição. Sem esse alinhamento, é pouco provável que a execução do resto do GQM tenha um impacto significativo. Para realizar a medição em termos precisos, cada meta GQM deve conter:

- Objetivo: O processo ou produto a ser estudado.
- Finalidade: Motivação por trás da meta (por que).
- Foco: Qualidade do objetivo em estudo (o que).
- Do ponto de vista: Perspectiva da meta (de quem).
- Ambiente: Da aplicação da medição.

Com o conhecimento dos itens anteriormentes descritos foi definido o objetivo de medição que pode ser visto na Tabela 6.

Tabela 6 – Objetivo de Medição (BERGHOUT; SOLINGEN, 1999)

ANALISAR:	CPU Spy
COM O PROPÓSITO DE:	Avaliar
COM RELAÇÃO A:	Desempenho
DO PONTO DE VISTA:	do usuário
NO CONTEXTO DE:	Fábrica de Software

5.1.2 Abstraction sheet

A extração das medidas necessárias para construção dos indicadores é comumente realizada por meio de entrevistas individuais dos membros do projeto. Estas entrevistas podem ser guiadas por diagramas chamados abstraction sheets na qual são estruturadas as informações a serem coletadas de forma que a equipe possua um ponto inicial para o refinamento das questões e métricas (BERGHOUT; SOLINGEN, 1999).

Apesar de neste TCC não serem realizadas extrevistas, o abstraction sheet foi escolhido para melhor visualização dos possíveis fatores que podem impactar a realização deste trabalho.

A Tabela 7 representa o abstraction sheet criado nesse trabalho:

Tabela 7 – Abstraction sheet (BERGHOUT; SOLINGEN, 1999)

FOCO DE QUALIDADE	**FATORES DE VARIAÇÃO**
Comportamento Temporal (CT)	Tempo de resposta
Utilização de Recursos (UR)	Recursos necessários para executar o software
HIPÓTESE DE BASELINE	**IMPACTO NAS HIPÓTESES DE BASELINE**
CT: Aceitável	Maior tempo de resposta piora o comportamento temporal
UR: Aceitável	Quanto mais recursos necessários para executa o software pior a utilização de recursos

5.1.3 Questões e Hipóteses

A transformação de objetivos em questões e hipóteses estabelece e aperfeiçoa a medição de metas, que se deslocam de um nível conceitual para um nível operacional, colocando algumas perguntas. Ao responder as perguntas, deve-se ser capaz de concluir se o objetivo foi alcançado. As perguntas foram derivadas do objetivo apresentado na Tabela 6 e servirão de base para determinar as métricas para cada objetivo definido.

1. Qual é o efeito no tempo de resposta para completar uma tarefa quando se utiliza uma técnica de desempenho?

2. Qual é o efeito na fatia de processamento utilizada pela aplicação quando se utiliza uma técnica de desempenho?

3. Qual é o efeito no consumo de memória da aplicação quando se utiliza uma técnica de desempenho?

5.1.4 Métricas

Nesta etapa, é examinada a forma como as questões poderiam ser respondidas. Neste trabalho, para cada questão elaborada foi definida uma métrica. A seguir estão apresentadas os nomes das métricas determinadas para este trabalho.

- M1 - Tempo de resposta
- M2 - Consumo de CPU
- M3 - Utilização máxima de memória

5.1.5 Detalhamento da Métrica

Cada uma das métricas deverá ser detalhada para facilitar a realização da coleta e análise dos dados.

A Tabela 8 detalha a métrica M2 apresentada na Seção 5.1.4:

Tabela 8 – Detalhamento de uma das Métricas

MÉTRICA	M2 – Consumo de CPU
DESCRIÇÃO	Percentual de uso de CPU pela aplicação
FÓRMULA	A ferramenta coleta a fatia de CPU que está em uso pela aplicação
TIPO DE ESCALA	Racional
ESTIDADES E ATRIBUTOS	**Entidade:** O software , **Atributos:** % de uso da CPU
TIPO DE MEDIÇÃO	Direta
CLASSIFICAÇÃO DA MEDIDA	Objetiva
FAIXA DE VALORES	0% a 100%
TIPO DE COLETA	Automática
COLETA	**Periodicidade:** a cada implementação de uma técnica, **Procedimento:** coleta automatizada, **Documentação:** gráfico de uso do CPU
ANÁLISE	**Responsável pela Análise:** Testador, **Periodo de Análise:** antes e após a implementação de uma técnica
APRESENTAÇÃO	**Forma de apresentação:** gráfica, **Interessado pela Informação:** Equipe de desenvolvimento
UNIDADE DE MEDIDA	%
METAS	Menos valor comparado às outras implementações

A Tabela 8 foi baseada no template para detalhamento das métricas apresentado por (RAMOS, 2013).

Em TCC 2 espera-se detalhar todas as métricas definidas na seção 5.1.4.

5.2 Coleta

A coleta foi realizada de forma automatizada,utilizando-se as ferramentas descritas no Capítulo 3. No entanto, identificou-se também a necessidade de um controle manual do tempo de monitoramento executado por uma das ferramentas (Little Eye) quando se desejar medir o tempo de resposta, pois tal monitoramento exige que o usuário clique nos botões para iniciar e parar a atividade. Para as medições foram executados os seguintes passos:

1. Aos trinta segundos, executar aplicação com a ferramenta Little Eye
2. Aos dois minutos, executar as funcionalidades Reset Timers e Refresh em sequência
3. Aos dois minutos e trinta segundos, executar a funcionalidade Refresh
4. Aos três minutos, executar a funcionalidade Refresh
5. Aos três minutos e trinta segundos, executar as funcionalidades Reset Timers e Refresh em sequência
6. Aos quatro minutos, executar a funcionalidade Refresh
7. Aos quatro minutos e trinta segundos executar a funcionalidade Refresh
8. Aos cinco minutos, parar monitoramento com a ferramenta Little Eye

5.3 Ambiente de Teste

Para execução dos testes, foi utilizado um smartphone e um notebook. A configuração de Hardware e Software de cada um pode ser observada abaixo.

- Smartphone: Samsung Galaxy SIII(I9300)
  - Configurações de Hardware:
    - Processador: Quad-core 1.4 GHz Cortex-A9
    - Memória: 16 GB
    - Memória RAM: 1 GB
  - Configurações de Software
    - Android: 4.3 (Jelly Bean)

- Notebook Samsung NP540
  - Configurações de Hardware:
    - Processador: Intel Core i3-370M
    - Memória: 4 GB DDR2
    - HD: 500 GB
  - Configurações de Software
    - Sistema Operacional: Microsoft Windos xP Professional
    - IDE: Eclipse Java Development Tools 3.8.2
    - Android: 4.4 (API 19)
    - Pacote Java : Java Development Kit 1.7

5.4 Resultados

Nesta seção será apresentada a técnica de programação escolhida para ser implementada, e medido o desempenho juntamente com o resultados obtidos.

5.4.1 Recursos utilizados

Foi implementada a técnica de programação "Usar sintaxe aprimorada da estrutura de repetição for" definida na Seção 2.4.3, foi utilizada a ferramenta little eye apresentada na Seção 3.2.2, para aferir o desempenho do software apresentado na Seção 3.3.1.

5.4.2 Resultados obtidos

O desempenho do software foi medido dez vezes antes da implentação da técnica de programação o resultado pode ser encontrado na Tabela 9:

Tabela 9 – Sem a implementação da técnica de programação

MÉTRICA	FÓRMULA	COLETA
Consumo de CPU	C = fatia do CPU utilizada	$C = 23 + 22 + 23 + 23 + 23 + 22 + 23 + 23 + 22 + 23/10$ C = 22,7%

Após a implementação da técnica, o software foi medido mais dez vezes. O resultado pode ser visualizado na Tabela 10

Tabela 10 – Com a implementação da técnica de programação

MÉTRICA	FÓRMULA	COLETA
Consumo de CPU	C = fatia do CPU utilizada	$C = 24 + 22 + 23 + 23 + 24 + 24 + 24 + 24 + 24 + 23/10$ C = 23,5%

Como pode ser visto nas Tabelas 9 e 10, o consumo de CPU aumentou com a utilização da técnica de programação, porém não se pode dizer com precisão que o desempenho do aplicativo piorou tendo em vista que é necessário detalhar e coletar as outras métricas também definidas na Seção 5.1.4 e aplicar em outros softwares.

Este experimento mostrou a viabilidade de se realizar a implementação das outras técnicas de programação utilizando as métricas para coletar os dados em TCC 2. Portanto, em TCC 2 serão detalhadas as outras métricas e realizadas outras medições.

6 CONSIDERAÇÕES FINAIS

Este capítulo apresenta as considerações finais deste Trabalho de Conclusão de Curso 1 e apresenta o que se espera realizar em Trabalho de Conclusão de Curso 2.

6.1 Conclusão

Motivado pela necessidade de aplicativos cada vez mais eficientes e pelo crescente número de dispositivos móveis com o uso do Sistema Operacional Android, o objetivo deste trabalho é avaliar o impacto de um conjunto de técnicas de programação no desempenho de um aplicativo. Após algumas pesquisas foram definidas as técnicas de programação; as métricas de software que serão utilizadas para medir o desempenho do software; as ferramentas que poderão ser utilizadas na segunda parte deste trabalho; bem como uma medição inicial que visou comprovar a viabilidade da execução do Trabalho de Conclusão de Curso 2.

O resultado da aplicação de uma das métricas detalhadas, mostrou uma pequena piora no consumo de CPU do software. Porém não se pode dizer ainda que o desempenho do software teve uma piora, pois o mesmo teste deverá ser aplicado em outro software e ser utilizadas as outras métricas que retornarão subsídios para uma conclusão precisa sobre a maneira como cada uma das técnicas de programação afeta o desempenho de uma aplicação desenvolvida utilizando a plataforma Android.

6.2 Trabalhos Futuros

Para a realização da segunda parte do trabalho, foi determinado um cronograma que pode ser visualizado na Seção 1.5. No Trabalho de Conclusão de Curso 2, pretende-se: coletar todos os dados referentes ao uso e desuso de cada técnica de programação definidas na Seção 2.4; procurar outras ferramentas para coleta de dados através do protocolo de pesquisa definido no Capítulo 3; detalhar as métricas que foram definidas, na Seção 5.1.4 .

Sendo assim, o objetivo final é realizar uma comparação precisa dos valores numéricos obtidos, mostrando de forma efetiva a maneira que as técnicas de programação afetam o desempenho de uma aplicação desenvolvida com o Sistema Operacional Android.

Referências

ANATEL. 2013. <http://www.anatel.gov.br/Portal/exibirPortalInternet.do>. Acessado em 24/03/2014. Citado na página 19.

BASILI, V. R.; CALDIERA, G.; ROMBACH, H. D. The goal question metric approach. *College Park. University of Maryland*, 1994. Citado 3 vezes nas páginas 13, 37 e 38.

BERGHOUT, E.; SOLINGEN, R. V. *THE GOAL/QUESTION/METRIC METHOD: a practical guide for quality improvement of software development*. [S.l.]: Company, 1999. Citado 4 vezes nas páginas 15, 37, 43 e 44.

BRERETON, P. et al. Lessons from applying the systematic literature review process within the software engineering domain. *5th International Workshop on Software and Performance*, 2006. Citado 2 vezes nas páginas 13 e 31.

DEVELOPER, G. *Developer Tools*. 2013. <http://developer.android.com/tools/index.html>. Acessado em 24/04/2014. Citado 2 vezes nas páginas 34 e 35.

ECLIPSE. 2013. <http://www.eclipse.org>. Acessado em 24/04/2014. Citado na página 35.

ENGHOLM, H. *Engenharia de Software na Prática*. 2010. <http://novatec.com.br/livros/engenhariasoftware/capitulo9788575222171.pdf>. Acesso em: 23/03/2014. Citado na página 21.

FSF. *Free Software Foundation*. 2004. <https://www.fsf.org/pt-br>. Acessado em 30/04/2014. Citado 2 vezes nas páginas 25 e 36.

GOOGLE. *Performance Tips*. 2013. <http://developer.android.com/training/articles/perf-tips.html>. Acessado em 10/03/2014. Citado na página 26.

HUB, G. *Git Hub*. 2014. <https://github.com/>. Acessado em 01/05/2014. Citado na página 36.

ISO/IEC. *Systems and software engineering – Measurement process*. 2007. Citado 3 vezes nas páginas 13, 38 e 39.

KOSLOSKI, R. A. D. *Notas de Aula da Disciplina Verificação e Validação de Software*. 2013. Disciplina ministrada no segundo semestre de 2013. Citado 2 vezes nas páginas 15 e 40.

MCGARRY, J. et al. *Practical Software Measurement: Objective Information for Decision MakersHardcover*. [S.l.]: ciencia, 2001. Citado 2 vezes nas páginas 39 e 41.

OHA. *Open Handset Alliance*. 2009. <http://www.openhandsetalliance.com/oha_overview.html>. Acessado em 20/03/2014. Citado na página 19.

PEREIRA, L. C. O.; SILVA, M. L. da. *Android para Desenvolvedores*. [S.l.]: BRASPORT, 2009. Citado 2 vezes nas páginas 19 e 20.

RAMOS, C. S. *Notas de Aula da Disciplina Medição e Análise*. 2013. Disciplina ministrada no primeiro semestre de 2013. Citado na página 45.

RANGARAJAN, K. et al. *Little Eye Labs*. 2013. <http://www.littleeye.co/>. Acessado em 25/04/2014. Citado 2 vezes nas páginas 35 e 36.

SILVA, E. L. da; MENEZES, E. M. *Metodologia da Pesquisa e Elaboração de Dissertação*. [S.l.]: Florianópolis, 2001. Citado na página 21.

YAMIN, A. et al. Explorando o escalonamento no desempenho de aplicações móveis distribuídas. *ciencia*, 2001. Citado na página 20.