UNIVERSIDADE DO VALE DO RIO DOS SINOS

Programa Interdisciplinar de Pós-Graduação em

Computação Aplicada

Mestrado Acadêmico

Alexsandro Souza Filippetto

gImpact: Uma Ferramenta para Análise de Impacto na Composição de Serviços para Dispositivos Móveis

Alexsandro Souza Filippetto

gImpact: Uma Ferramenta para Análise de Impacto na Composição de Serviços para Dispositivos Móveis

Dissertação apresentada à Universidade do Vale do Rio dos Sinos como requisito parcial para obtenção do título de Mestre em Computação Aplicada.

Orientador: Prof. Dr. Sérgio Crespo Coelho da Silva Pinto

São Leopoldo, 2009.

F483g Filippetto, Alexsandro Souza

gImpact: uma ferramenta para análise de impacto na composição de serviços para dispositivos móveis/ por Alexsandro Souza Filippetto. -- 2009.

112 p. : il. ; 30cm.

Dissertação (mestrado) -- Universidade do Vale do Rio dos Sinos, Programa de Pós-Graduação em Computação Aplicada, São Leopoldo, RS, 2009.

"Orientação: Prof. Dr. Sérgio Crespo Crespo da Silva Pinto, Ciências Exatas e Tecnológicas".

1. Computação Móvel - Análise de Impacto. 2. Arquitetura Orientada a Serviços I. Título.

CDU 004.75.057.5

Catalogação na Publicação:
Bibliotecária Eliete Mari Doncato Brasil - CRB 10/1184

Dedico este trabalho às pessoas mais importantes da minha vida: meus pais Renato e Marisa, meus irmãos Adriano e Elisandro e a minha namorada Camila.

AGRADECIMENTOS

A Deus por todas as dádivas recebidas nessa vida, por permitir que convivesse ao lado de grandes pessoas as quais posso chamar de amigos, pelas oportunidades concedidas e pelo dom da vida.

Aos meus queridos pais, Renato e Marisa, pelo exemplo de vida que sempre tive e pelo incentivo incondicional que sempre me propuseram.

Aos meus irmãos Adriano e Elisandro, pessoas muito especiais, por seu apoio e carinho sinceros.

À minha amada namorada Camila, por seu amor, compreensão, por todo apoio nos momentos difíceis e por saber compreender as privações e ausências decorrentes do mestrado.

Ao professor Sérgio Crespo pela orientação, confiança e incentivo; por conduzir meu desenvolvimento com muita sabedoria e paciência.

Aos professores André Raabe e Jorge Barbosa por sua participação na banca e por suas valiosas contribuições no refinamento do trabalho.

Aos meus colegas do projeto U-SOA, Luciano Zanuz e Giovane Barcelos, pelo companheirismo e ajuda durante o projeto.

Aos alunos do curso de Engenharia da Computação, Cládio, Douglas e João, que compraram a idéia e nos auxiliaram no projeto U-SOA.

Aos professores e funcionários do PIPCA que sempre que foram solicitados deram a sua valorosa contribuição.

Aos meus amigos que souberam compreender a minha ausência.

A CAPES pelo apoio financeiro.

E a todos aqueles que direta ou indiretamente contribuíram para a realização deste trabalho.

“Nenhum problema pode ser resolvido pelo mesmo estado de consciência que o criou. É preciso ir mais longe. Eu penso várias vezes e nada descubro. Deixo de pensar, mergulho em um grande silêncio e a verdade me é revelada."

Albert Einstein

RESUMO

O uso intenso de serviços na Web tem se difundido muito com o surgimento de aplicações baseadas em Web-Services. A evolução natural foi o aparecimento de arquiteturas orientadas a serviços, chamadas de SOA. Estas arquiteturas oferecem recursos para se trabalhar a ideia de composição de serviços por meio de orquestração ou coreografia. Desta forma, um usuário pode usar ou construir um serviço na Web compondo com outros já existentes. Esta ideia em dispositivos móveis ainda é recente e requer produtos e ferramentas como a otimização das composições visando sempre obter o melhor custo/benefício. Nesta perspectiva, é importante oferecer ao usuário um ferramental onde ele possa ter ciência da complexidade e obter índices de qualidade dos serviços que ele pretende construir.

Com isso surge a necessidade de se monitorar o impacto e qualidade dos serviços oferecidos através de dispositivos móveis, para que seja possível a seleção da melhor composição para atendimento a um requisito sem que se altere o significado da composição original. Neste contexto, a dissertação apresenta a ferramenta gImpact, que visa fornecer através de serviços um conjunto de ferramentas para avaliação do impacto e qualidade dos serviços dispostos nos dispositivos móveis ou em *desktops*.

Palavras-chave: Análise de Impacto; Arquitetura Orientada a Serviços; Computação Móvel.

ABSTRACT

The intense use of services in the Web has become very widespread with the applications emergence based on Web-Services. The natural evolution was the arising of services oriented architecture, called SOA. These architectures provide resources to work the composition idea through orchestration or choreography. Thus, a user can use or build a web service composing it with other existing ones. This idea in mobile devices is still recent and requires products and tools such as compositions optimization to always get the best cost/benefits. In this perspective, it is important provide to the user a tool where he can take science of complexity and obtain rates of services quality that it intends to build.

That fact brings up the need of monitoring the impact and quality of services offered by mobile devices,it would enable the selection of the best composition to meet a requirement, without changing the meaning of the original composition.In this context, the thesis presents the gImpact tool, which aims to provide services through a set of tools to evaluate the impact and quality of services in desktops or mobile devices..

Keywords**:** *Impact Analysis; Service-Oriented Architecture*; *Mobility Computing.*

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Program Interface*
BD	Banco de Dados
BPEL	*Business Process Execution Language*
CDC	*Connected Device Configuration*
CIL	*Common Intermediate Language*
CLDC	*Connected Limited Device Configuration*
CLR	*Common Language Runtime*
CPU	*Computer Process Unit*
DDS	Desenvolvimento Distribuído de *Software*
DOM	*Document Object Model*
ECMA	*European Computer Manufacturer`s Association*
FCL	*Framework Class Library*
HTML	*HyperText Markup Language*
JEE	Java *Enterprise Edition*
JIT	*Just-In-Time*
JME	Java *Micro Edition*
JSE	Java *Stsndard Edition*
kSOAP	*k-Simple Object Access Protocol*
kXML	*k-Extensible Markup Language*
MIDP	*Mobile Information Device Profil*e
MVC	*Model-View-Controller*
PDA	*Personal Digital Assistent*
QoS	*Quality of Service*
SAX	*Simple* API *for* XML
SDK	*Software Development Kit*
SGML	*Standard General Markup Language*
SLA	*Service Level Agreement*
SOA	*Service Oriented Architecture*
SOAP	*Simple Object Access Protocol*
SPC	Serviço de Proteção ao Crédito
UDDI	*Universal Description, Discovery and Integration*
UML	*Unified Modeling Language*
W3C	*Word Wide Web Consortium*

WS	*Web Services*
WSAL	*Web Services Level Agreement*
WSDL	*Web Services Description Language*
WSML	*Web Services Management Language*
WSOL	*Web Services Offer Language*
XML	*Extensible Markup Language*

1 INTRODUÇÃO

Mark Weiser descreveu, em 1991, os princípios de uma visão para a computação ubíqua, além de alguns cenários que futuramente utilizariam este novo conceito [WEISER, 1991]. Anos mais tarde, Satyanarayanan [SATYANARAYANAN, 2001] definiu que uma das visões da computação ubíqua é a criação de ambientes totalmente computacionais com grande capacidade de comunicação, que abarcam, de uma forma inteligente, a computação com usuários humanos. Desde então, diversas pesquisas são realizadas nesta abrangente área e um conceito que emerge na utilização de dispositivos móveis é o de ambientes colaborativos, nos quais diversas pessoas, ou até mesmo recursos, colaboram entre si para alcançarem um determinado objetivo. Este cenário foi acentuado com arquiteturas interoperáveis que possam utilizar *web services* em conjunto com ontologias para permitir a integração de diferentes aplicações [THOMAS et al., 2003].

Com isso, uma arquitetura possível para implementação destes conceitos seria orientada a serviços, uma vez que dispositivos móveis possuem normalmente uma capacidade menor de processamento. Nesse sentido, a possibilidade de se transferir o processamento para uma máquina de maior poder computacional é uma alternativa. Assim, SOA (*Service Oriented Architecture*) se mostra uma opção para a utilização de serviços e colaboração entre os diversos usuários.

Service Oriented Architecture (SOA) tem como princípio o uso de serviços para dar suporte aos requisitos de negócio. Estes serviços são funções que podem ser utilizadas como uma unidade de *software*, normalmente implementada através de *Web Services*, embora possa também ser executada com outras tecnologias. A arquitetura orientada a serviços visa a promover a junção entre componentes de *software*, de forma que eles possam ser usados como um novo componente.

Neste cenário, cuja unidade de *software* é um serviço, surge a necessidade de se analisar os impactos e a qualidade de uma composição, uma vez que o enfoque orientado a serviços simplifica as comunicações entre sistemas a um ponto tal que não se sabe se um determinado serviço reside nos seus próprios computadores ou em outros servidores. Com isto, o aumento da complexidade dos serviços é essencial a validação e otimização em sua utilização [HP SOA, 2008]. Neste contexto, uma solução proposta para seleção dos serviços para uma composição seria através da análise de impacto. Para isso, é necessária a criação de

ferramentas que agreguem informações referentes a este cenário para que seja possível um gerenciamento destas composições ou serviços.

O trabalho de [BRIAND et al., 2006], define análise de impacto como o processo de estimar as potenciais consequências de uma alteração ou da complexidade de uma unidade de *software*, seja ela uma função, um programa ou um serviço. Neste trabalho, a análise de impacto em uma composição consiste em identificar quais os serviços que fazem parte de composição e através de uma fórmula proposta estimar um custo para a mesma, assim permitindo a seleção dos serviços com menor custo para a composição, visando sua otimização.

A ferramenta para análise de impacto, denominada gImpact, faz parte de um projeto maior de nome U-SOA (*Ubiquitous SOA Framework*) [ZANUZ et al., 2008], de desenvolvimento de um grupo de pesquisa da universidade, onde alguns de seus componentes são necessários para a concepção do gImpact. A linguagem para composição em dispositivos móveis (inSOA), é utilizada para a elaboração das composições, o que permite que sejam consultados os serviços que fazem parte de uma composição, permitindo assim uma análise sobre os mesmos. Enquanto que se faz necessário um motor para execução das composições nos dispositivos, SmallSOA [ZANUZ, FILIPPETTO et al., 2008]. Estes dois trabalhos fazem parte do projeto U-SOA, e são utilizados pela ferramenta gImpact.

1.1 Motivação

Com a popularização dos dispositivos móveis e sua utilização cada vez maior na indústria como um todo, torna-se cada vez mais indispensável a criação de ambientes que utilizem todo o potencial destes dispositivos. Neste sentido, a utilização de dispositivos móveis, acessando e provendo serviços, torna maior a necessidade de uma análise constante sobre a qualidade dos serviços disponíveis e sobre o impacto nas composições, uma vez que ao se utilizar um serviço não se sabe se este é uma composição de outros serviços ou seria um único.

Com isso, a principal motivação deste trabalho é permitir que seja possível um gerenciamento sobre as composições e prover informações quanto à sua disponibilidade e qualidade, permitindo, assim, uma tomada mais ágil de decisões sobre mudanças em serviços.

1.2 Definição do Problema

Uma das características da Internet diz respeito à volatilidade dos serviços oferecidos. Devido ao ambiente de rede ser altamente dinâmico, não é possível garantir a estabilidade de uma conexão entre os nodos de uma rede [FOSTER 2002]. Neste sentido, a execução de uma composição, com serviços que estejam em servidores diferentes, pode ser comprometida por esse dinamismo nos serviços e/ou até mesmo deixar de ser uma composição válida por falta de um serviço que fazia parte da composição e deixou de existir.

Outra questão é que, em geral, o desenvolvimento de *software* comercial responde rapidamente a mudanças e evoluções tecnológicas. Assim, *softwares* sofrem constantes evoluções [SOMMERVILLE, 2004], desde alterações estruturais, como alterações em suas funcionalidades. Em uma arquitetura orientada a serviços, onde "funções" são expostas na Internet e não se sabe sua estrutura, uma questão importante é permitir a análise destas composições. Além de identificar, de forma precisa, quais outros serviços fazem parte de uma composição.

1.3 Objetivos

O principal objetivo deste trabalho é especificar e implementar uma ferramenta voltada a dispositivos móveis que permita realizar uma análise de impacto em composições e fornecer informações quanto à disponibilidade e qualidade dos serviços que fazem parte de uma determinada composição.

Com isso, este trabalho tem como objetivos:

- Estudar o estado da arte de pesquisas relacionadas à análise de impacto em serviços;
- Pesquisar e utilizar as tecnologias relacionadas a serviços como *web services* e plataformas para implementação em ambientes móveis;
- Definir a arquitetura geral com os pontos para integração com trabalhos relacionados com o grupo de pesquisa;
- Projetar e implementar os artefatos necessários para prover a análise de impacto em composições;

- Definir a arquitetura para a análise de disponibilidade e qualidade de todos os serviços que fazem parte de uma composição, fornecendo, assim, dados de toda a árvore de composições;
- Construir um protótipo para avaliar a ferramenta.

1.4 Organização da Dissertação

Esta proposta possui sete capítulos, sendo que, no primeiro, encontra-se a introdução. Os demais capítulos são descritos a seguir:

- Capítulo 2: Tecnologias Relacionadas – descreve as tecnologias utilizadas ao longo do desenvolvimento do trabalho. O trabalho possui como foco as tecnologias voltadas a serviços, como SOA, *Web Services*, conceitos sobre Análise de Impacto, e as plataformas possíveis para o desenvolvimento, como Android, Java ou .NET;
- Capítulo 3: Trabalhos Relacionados – apresenta alguns trabalhos com assuntos relacionados a esta proposta, abordando, desde a composição de serviços, até a análise de impacto na área tecnológica;
- Capítulo 4: Implementação do Projeto – descreve a implementação da ferramenta para análise de impacto através de metodologias e padrões de mercado como a UML e o modelo MVC para desenvolvimento;
- Capítulo 5: Estudos de Caso – apresenta três cenários com a utilização da ferramenta desenvolvida. Dois dos cenários visam demonstrar sua utilização na análise de impacto, enquanto que o último cenário apresenta a reutilização dos componentes desenvolvidos em uma ferramenta de *profiling*;
- Capítulo 6: Conclusões – neste capítulo são apresentadas as conclusões do trabalho desenvolvido;
- Capítulo 7: Referências Bibliográficas – lista as fontes de pesquisa utilizadas na realização deste trabalho.

2 TECNOLOGIAS RELACIONADAS

Este capítulo descreve uma visão geral das principais tecnologias aplicadas neste trabalho, tais como: SOA, *Web Services* e Plataformas para desenvolvimento em dispositivos móveis.

2.1 Service Oriented Architecture

Service Oriented Architecture (SOA) é uma abordagem de desenvolvimento de *software* na qual suas funções ou serviços são construídos como componentes reutilizáveis. Esta abordagem visa a fornecer um baixo acoplamento, além de interoperabilidade, habilidade de descobrimento, gerenciamento de alterações e operação de serviços de negócio em um ambiente bem administrado. Serviços de negócio, operando em um ambiente SOA, podem ser compostos por processos de negócio que alinham TI com o negócio [PIJANOWSKI, 2007].

O princípio de SOA é o uso de serviços para dar suporte os requisitos de negócio [HIGH, KINDER e GRAHAM, 2005]. Por serviços, entende-se basicamente como funções que possam ser utilizadas como uma unidade, normalmente implementadas através de *Web Services*, embora a arquitetura SOA possa ser implementada com outros protocolos e tecnologias [SAMPAIO, 2006]. A seguir, a Figura 2.1 apresenta um modelo para a arquitetura SOA, demonstrando serviços que possam ser executados em computadores diferentes, nos quais estes serviços são orquestrados em um ponto específico para apresentação dos resultados ao final de suas execuções.

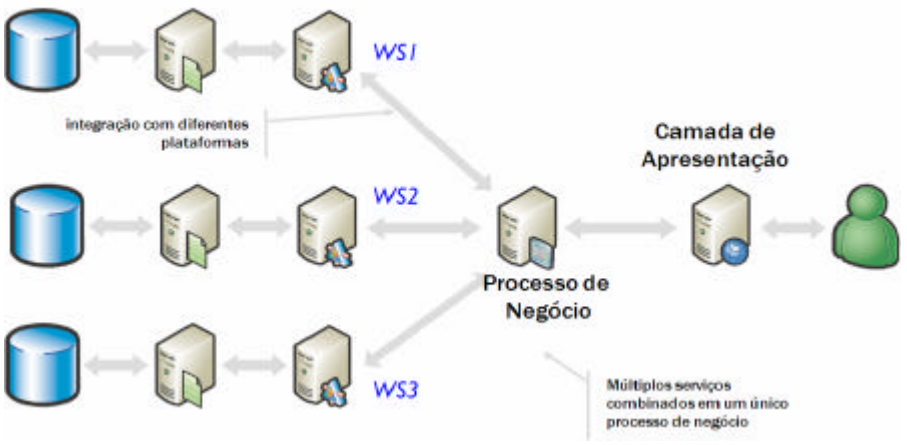

Figura 2.1 - Representação de Serviços

SOA é uma arquitetura que visa oferecer e consumir *software* como serviços. Há três entidades que compõem a arquitetura SOA: 1) os provedores de serviço; 2) requisitores de serviço (também conhecidos como consumidores de serviço); e 3) registro de serviço. Os provedores de serviço são os donos que oferecem serviços, pois definem descrições de seus serviços e os publicam no registro de serviço. Os consumidores de serviços usam uma operação de busca para localizar serviços de seu interesse. Já o registro devolve a descrição de cada serviço pertinente. Assim, o consumidor usa esta descrição para invocar o serviço correspondente [FALKL, 2005].

Atualmente, a tecnologia mais utilizada para a implementação da arquitetura SOA tem sido *Web Services*, onde implementa SOA por meios de uma padronização de XML. Nesta abordagem, três iniciativas são usadas para apoiar interações entre *Web Services*: SOAP (um modo para comunicação), WSDL (um meio para a descrição de serviços) e UDDI (um nome e servidor de diretório). A seguir, a figura 2.2 ilustra o paradigma de SOA com as tecnologias mencionadas [SÁNCHEZ-NIELSEN et al., 2006].

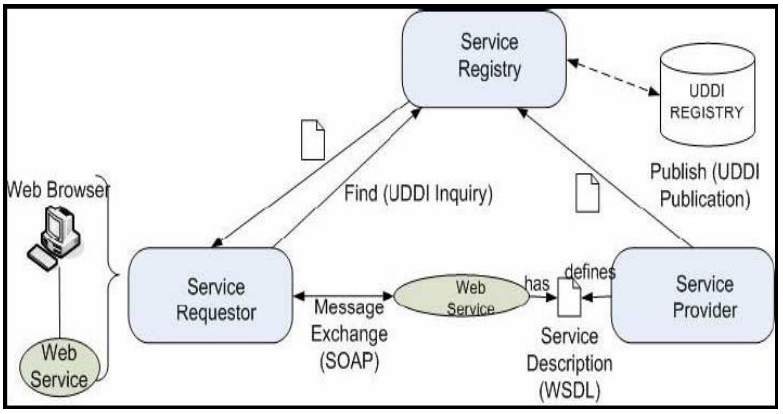

Figura 2.2 - Arquitetura SOA

A arquitetura SOA é vista com um paradigma ideal para integrar ambientes heterogêneos, pois sua unidade de desenvolvimento são os serviços. Assim, estes podem ser executados em ambientes distintos [HARRISON e TAYLOR, 2006].

2.1.1 Serviços

Um serviço é um componente que atende a uma função de negócio específica para seus clientes, pois recebe requisições e as responde ocultando todo o detalhamento de seu

processamento [SAMPAIO, 2006]. Além disso, os serviços são as unidades principais de SOA, sendo que SOA pode ser considerada como uma coleção e orquestração de serviços [BIH, 2006]. Serviços são unidades de trabalho criadas por um fornecedor de serviço para alcançar resultados finais desejados por um de seus consumidores. Simplificando, um serviço é uma tarefa repetível dentro de um processo de negócio.

Outra definição de serviço diz respeito à lógica de negócio (servidor), isto é, ele é capaz de ser acessado por um outro processo (cliente). Tipicamente, o cliente é um outro processo executando em outra máquina que está utilizando a rede para acessar o servidor [PIJANOWSKI, 2007].

Nesse aspecto, embora os serviços originais, tais como DCOM (*Distributed Component Object Model*) e CORBA (*Common Object Request Broker Architecture*), ainda possam ser utilizados, a tecnologia de *Web Services* tem sido a mais utilizada na implementação de SOA como os serviços de arquitetura, ainda que a combinação de tecnologias seja possível.

2.1.2 Orquestração

Na orquestração, um processo central controla os serviços e coordena a execução de diferentes operações nos serviços envolvidos no processo. Sendo assim, os serviços não sabem e não precisam saber que estão envolvidos em um processo de composição e também que estão fazendo parte um processo de negócio de nível mais alto. A Figura 2.3 apresenta um esquema de orquestração de quatro serviços disponibilizados através de *web services*.

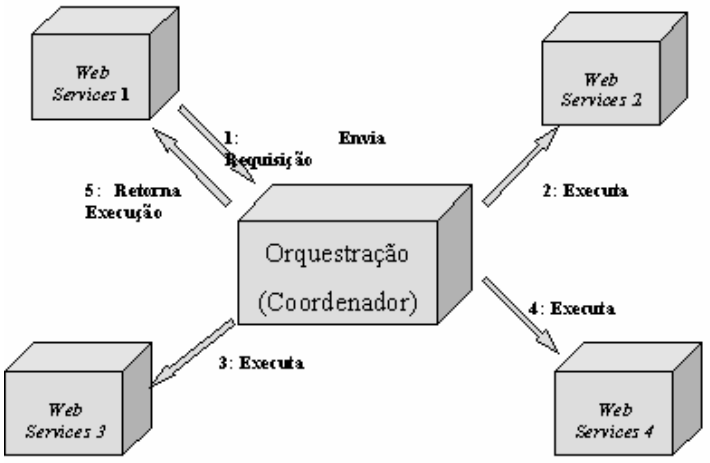

Figura 2.3 - Composição de Serviços com Orquestração

A orquestração é composta por um fluxo de etapas, com verificações de pré-condições e pós-condições, e um coordenador, responsável por dar um andamento ao fluxo. O cliente se comunica com o coordenador e efetua as solicitações de serviços, enquanto o coordenador inicia o fluxo, executando e verificando todas as etapas necessárias. Deste modo, pode-se mudar a ordem das etapas ou acrescentar novas verificações ou até mesmo acrescentar novos serviços, sem que isso afete os serviços já desenvolvidos [SAMPAIO, 2006].

2.1.3 Web Services

Segundo [CRESPO, 2000], "um serviço dentro da *Web* é composto por um ou vários *scripts* que realizam uma determinada tarefa a partir do *browser*. Geralmente serviços na *Web* podem ser vistos como caixas pretas onde os parâmetros passados pelo *browser* são decodificados por um *script* no Servidor *WWW* da aplicação."

Fracamente acoplados, *Web Services* são aplicativos que disponibilizam um ou mais serviços para seus clientes [SAMPAIO, 2006]. Desta forma, são aplicações modulares que podem ser publicadas, localizadas e executadas via *web*. *Web Services* podem executar funções através de uma simples requisição para compor processos mais complexos [THOMAS et al., 2003], uma vez que um *Web Services* executa uma função específica e não necessita que o cliente que o executa conheça o código de tal execução, devendo apenas conhecer os parâmetros de entrada e saída. A Figura 2.4 representa um cliente executando um *Web Services* através de um protocolo de comunicação.

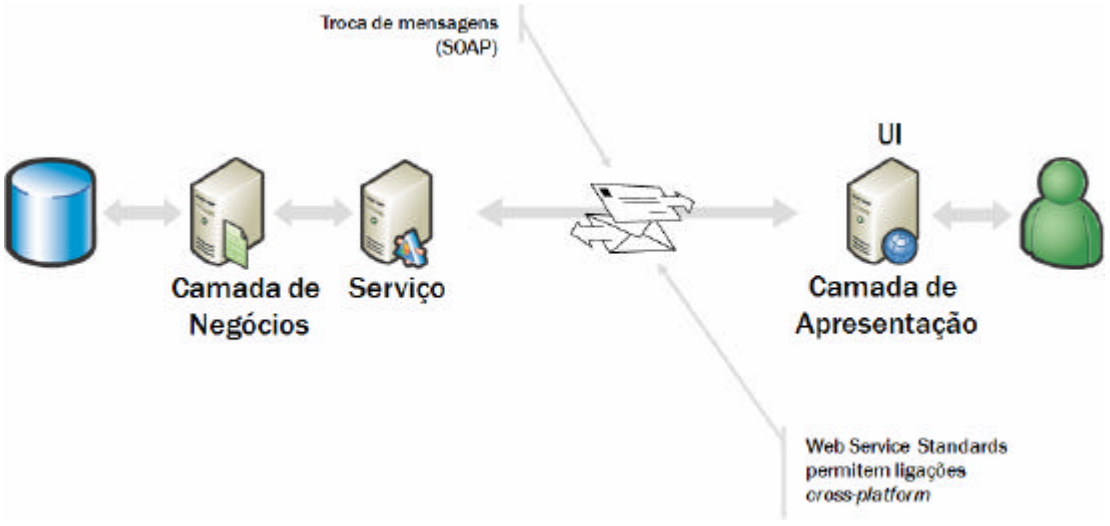

Figura 2.4 - Modelo de *Web Services*

O protocolo de comunicação mais utilizado para a execução de *Web Services* tem sido SOAP, que é um protocolo de transmissão de mensagens baseado em XML. SOAP é um padrão recomendado pela *World Wide Web Consortium* (W3C), a qual define um conjunto de regras para construção das mensagens que são utilizadas na comunicação [SHIL et al., 2006].

A linguagem para descrição de um *Web Services* é a WSDL (*Web Services Description Language*). Uma descrição de serviço WSDL dá conta do ponto do contato de um serviço, também conhecido como *service endpoint*. Ele estabelece a localização física do serviço e fornece uma definição formal da sua interface, de forma que os programas que desejam se comunicar com os *Web Services* possam saber exatamente como estruturar as mensagens de requisição necessárias. A Figura 2.5 apresenta os principais artefatos envolvidos em uma execução de um *Web Services*.

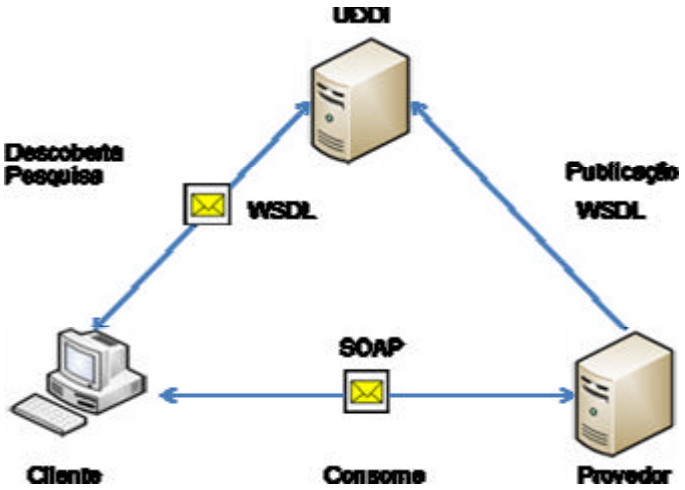

Figura 2.5 – Execução de *Web Services*

O provedor é a plataforma acessada na solicitação do serviço. Trata-se da entidade que cria o *Web Services* sendo responsável por fazer sua publicação em formato padrão e publicá-lo. O registro de serviço, servidor onde são publicados os serviços conhecido como UDDI (*Universal Description, Discovery and Integration*), é o local onde os provedores publicam as descrições dos serviços. O funcionamento inicia quando o provedor cria uma descrição que detalha a interface do serviço, ou seja, suas operações e as mensagens de entrada e saída para cada operação. Dessa forma, uma descrição de ligação é criada, mostrando, então, como enviar cada mensagem para o endereço onde o *Web Services* está localizado. Quanto ao cliente, ele é uma aplicação que invoca ou inicia uma interação com um determinado serviço.

2.1.4 eXtensible Markup Language

XML (*eXtensible Markup Language*) é uma linguagem de marcadores como a HTML (*HyperText Markup Language)* e foi desenhada para descrever dados. Sua maior vantagem é que ela é extensível, ou seja, essa linguagem não se limita a um certo número de tags. Além disso, pode criar as suas próprias tags. Nesse sentido, XML é uma linguagem autodefinível [SAMPAIO, 2006].

Em 1996, a *World Wide Web Consortium* (W3C) lançou uma linguagem de marcação que combinasse flexibilidade e simplicidade, neste contexto surge o XML. O princípio do projeto era criar uma linguagem que pudesse ser lida por *software* e que pudesse ser facilmente interpretada. A seguir, a Figura 2.6 apresenta um exemplo de código XML. Para tornar o XML uma linguagem simplificada, alguns requisitos foram necessários. São eles:

- Separação do conteúdo da formatação;
- Simplicidade e Legibilidade;
- Possibilidade de criação de tags;
- Criação de arquivos para validação de estrutura através de um arquivo DTD (*Document Type Definition*);
- Interligação de bancos de dados distintos;
- Concentração na estrutura da informação.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
- <pessoas>
- <pessoa>
<nome>Fulano</nome>
<email>ful@ano</email>
<idade>20</idade>
</pessoa>
- <pessoa>
<nome>Ciclano</nome>
<email>blabla@eeee.com</email>
<idade>56</idade>
</pessoa>
- </pessoas>
```

Figura 2.6 – Estrutura Básica de um Documento XML

XML é uma linguagem para organização, e não para apresentação, dos dados de um documento. Um documento XML é um documento em formato textual que contém marcadores. Esses marcadores são definidos através de uma linguagem onde sua sintaxe é baseada em marcas (tags) [SAMPAIO, 2006]. Para a utilização de tags, não existe um padrão definido, o que torna o XML uma linguagem flexível que pode atender a diferentes propósitos.

Os dados em um documento XML podem ser representados de duas maneiras diferentes [SAMPAIO, 2006]:

- Elementos: os elementos podem conter dados vazios, texto, outros elementos, ou texto com outros elementos. O nome de um elemento deve ser único para representar um determinado tipo de dado;
- Atributos: um atributo serve para fornecer uma informação adicional sobre o elemento a qual ele se refere.

Com a finalidade de ser uma linguagem de interpretação mais simples, a linguagem XML foi criada como um subconjunto da *Standard General Markup Language* (SGML), uma vez que SGML é um padrão complexo para descrever a estrutura do conteúdo de documentos.

2.1.5 k-eXtensible Markup Language

O kXML é responsável por dar suporte à linguagem XML. Trata-se de uma biblioteca de componentes responsáveis pela manipulação de documentos XML. Esta biblioteca se tornou popular, pois seu desenvolvimento foi voltado para ambientes de dispositivos móveis [kXML, 2008]. Por isso, a base para a construção da biblioteca kXML foram os *parsers* comuns de XML, como SAX (*Simple* API *for* XML) e DOM (*Document Object Model*). As principais características herdadas do SAX e DOM são citadas a seguir [kXML, 2008]:

- O kXML permite, assim como a biblioteca SAX, delegar a leitura de um documento XML a um método que execute o *parser* no arquivo, gerando determinados eventos para tratamento;
- É possível a leitura de um documento da mesma forma utilizada em DOM, carregando todo o documento para a memória e manipulando suas partes através de elementos e atributos;

- Há uma diferença em relação à manipulação através do DOM, o *parser* kXML permite que a árvore seja manipulada estando essa parcialmente carregada em memória.

2.1.6 Web Service Description Language

Web Service Description Language (WSDL) define um padrão para a descrição de serviços. Através dela, descrevemos as interfaces para os serviços externos, ou serviços que são oferecidos por uma determinada aplicação, independente de sua plataforma ou linguagem de programação.

WSDL separa a descrição dos serviços em duas partes [SHIL et al., 2006]. Nesse sentido, a descrição abstrata estabelece as características de interface do *Web Services* sem qualquer referência às tecnologias usadas para hospedar ou transmitir as mensagens. Separando essa informação, a integridade da descrição do serviço pode ser preservada independentemente das alterações que possam ocorrer na plataforma tecnológica. Já a descrição concreta define o protocolo de transporte físico para possibilitar que a interface abstrata do *Web Services* possa comunicar-se. Assim, WSDL provê uma descrição simples para a execução de um *Web Services* e de fácil consulta.

O principal objetivo de um WSDL é descrever as interfaces apresentadas dos serviços e fornecer a localização dos seus serviços, disponíveis em um determinado local na rede. Por ser um documento XML, sua leitura se torna fácil e acessível. Um documento WSDL se divide em elementos, conforme apresenta a Figura 2.7, que segue:

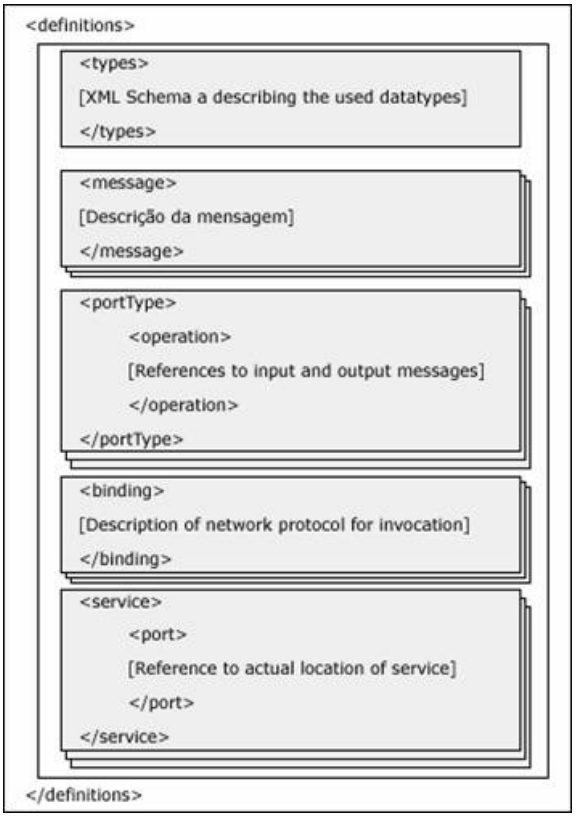

Figura 2.7 – Elementos de um Documento WSDL

Através destes elementos, é possível uma maior flexibilidade na utilização do WSDL, já que estes podem ser reutilizados para definir diferentes tipos de serviços. Segundo Sampaio (2006), os principais elementos apresentados são:

- **<types>**: O elemento types contém os tipos de dados que estão presentes na mensagem;
- **<message>**: Define os parâmetros de entrada e saída de um serviço. Cada elemento *message* recebe um ou mais elementos do tipo <part>, os quais formam as partes reais da mensagem;
- **<operation>**: Demonstra o relacionamento entre parâmetros de entrada e de saída, apresentando a assinatura do método;
- **<portType>**: Apresenta um agrupamento lógico de operações <operation>;
- **<binding>**: Define o protocolo a ser usado para acessar os métodos de um objeto (SOAP, HTTP ou MIME);

- <**service**>: Expõe o endereço do serviço.

Para aumentar a possibilidade de utilização de elementos em um documento WSDL, por se tratar de uma descrição em XML, pode-se empregar *Namespaces*, utilizando-se de atributos para fazer referência a outros elementos, seja dentro ou fora do documento. A seguir, o Quadro 2.1 apresenta alguns dos principais *namespaces* utilizados em um WSDL.

Quadro 2.1 – Namespaces de um Documento WSDL

Prefixo	**URI do Namespace**	**Descrição**
WSDL	http://schemas.xmlsoap.org/wsdl	Namespace de WSDL para framework WSDL
SOAP	http://schemas.xmlsoap.org/wsdl/soap	Namespace de WSDL para vínculo de SOAP e WSDL
HTTP	http://schemas.xmlsoap.org/wsdl/http	Namespace de WSDL para WSDL http GET & vínculo POST
Mime	http://schemas.xmlsoap.org/wsdl/mime	Namespace de WSDL para vínculo MIME de WSDL
Soapenc	http://schemas.xmlsoap.org/soap/encoding	Namespace de codificação conforme definido pelo SOAP 1.1
Soapenv	http://schemas.xmlsoap.org/soap/envelope	Namespace de codificação conforme definido pelo SOAP 1.1
XSI	http://www.w3.org/2001/XMLSchema-instance	Namespace da instância conforme definido pelo esquema de XML
XSD	http://www.w3.org/2001/XMLShema	Namespace do esquema conforme definido pelo esquema de XML

A Figura 2.8 apresenta um trecho de código contendo um exemplo de documento WSDL com os elementos citados anteriormente.

```
      <?xml version="1.0" encoding="ISO-8859-1" ?>
      <message name="helloRequest">
            <part name="name" type="xsd:string" />
      </message>
      <message name="helloResponse">
            <part name="return" type="xsd:string" />
      </message>
      <portType name="server.helloPortType">
            <operation name="hello" />
      </portType>
      <service name="server.hello">
            <port name="server.helloPort"
binding="tns:server.helloBinding">
            <soap:address location="http://localhost/server" /> </port>
      </service>
      </definitions>
```

Figura 2.8 – Exemplo de Documento WSDL

Neste exemplo, é possível notar que os elementos *message* estão indicando os parâmetros para a chamada do serviço, tanto de entrada quanto de saída. O *portType*, por sua vez, indica quais operações serão executadas. E, por fim, o *service* indica o serviço que será executado e seu caminho.

2.1.7 Simple Object Access Protocol

Simple Object Access Protocol, ou simplesmente SOAP, foi criado para possibilitar a invocação remota de métodos através da Internet. Para isso, utilizaram o protocolo mais difundido de comunicação na Internet, o HTTP, e uma forma de comunicação já amplamente difundida que permitisse uma flexibilidade, porém, que fosse padronizada, o XML. Atualmente, SOAP é um protocolo de comunicação padronizado e regulado pela W3C [SAMPAIO, 2006]. SOAP permite a comunicação em um universo heterogêneo, uma vez que é baseado em XML e permite sua manipulação em diversas plataformas.

Além do mais, SOAP é um protocolo para troca de informações estruturadas em uma plataforma descentralizada e distribuída. Utilizando tecnologias baseadas em XML, chama-se de Envelope a comunicação enviada através de SOAP. A Figura 2.9 apresenta uma estrutura padrão de um envelope SOAP. Sua especificação define um *framework* que provê maneiras para se construir mensagens que podem trafegar através de diversos protocolos e que foi especificado de forma a ser independente de qualquer modelo de programação [SOAP, 2008].

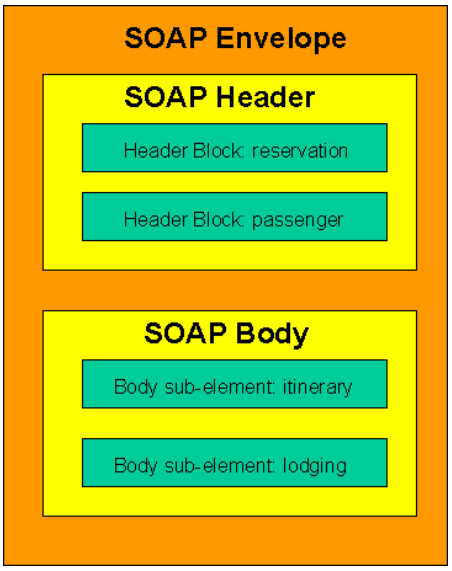

Figura 2.9 – Estrutura de uma Mensagem SOAP [SOAP, 2008]

Segundo Sampaio (2006), as mensagens SOAP são divididas basicamente em três elementos principais:

1. Envelope: contém o *header* e o *body* e marca o início e o final da mensagem;
2. *Header*: são informações de controle da mensagem XML, normalmente informações para controle. Em uma mesma mensagem, podem existir mais de um *header*, sendo que alguns podem ser obrigatórios e outros opcionais;
3. *Body*: contém o corpo da mensagem, tanto a que está sendo enviada, quanto a que está sendo recebida. Se a execução do serviço for correta, virá, então, sua resposta no corpo da mensagem. Caso tenha ocorrido um erro, virá no corpo um elemento do tipo *Fault* contendo a descrição do erro.

Toda a mensagem SOAP é formada obrigatoriamente por um elemento Envelope que contém subelementos, *header* e *body*, como apresenta a figura. Um exemplo de uma mensagem SOAP é apresentado na Figura 2.10.

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope" >
 <env:Header>
     <t:transaction
        xmlns:t="http://thirdparty.example.org/transaction"
          env:encodingStyle="http://example.com/encoding"
           env:mustUnderstand="true">5</t:transaction>
 </env:Header>
 <env:Body>
     <m:chargeReservationResponse
         env:encodingStyle="http://www.w3.org/2003/05/soap-encoding"
             xmlns:m="http://travelcompany.example.org/">
       <m:code>FT35ZBQ</m:code>
       <m:viewAt>
         http://travelcompany.example.org/reservations?code=FT35ZBQ
       </m:viewAt>
     </m:chargeReservationResponse>
 </env:Body>
</env:Envelope>
```

Figura 2.10 – Exemplo de uma Mensagem de Resposta SOAP

Neste exemplo, existe a tag “Envelope”, onde foi atribuído um *namespace* de “env”, e seus subelementos, *header*, com informações de configuração, e o *body*, contendo o retorno da execução do método na tag “code”.

2.1.8 k-Simple Object Access Protocol

Assim como o kXML, kSOAP é uma biblioteca desenvolvida para ambientes móveis ou com um menor poder de armazenamento e processamento, isto é, as que utilizam a plataforma JME (atendendo a dispositivos do tipo CLDC, CDC, MIDP) [kSOAP, 2008].

À versão atual do kSOAP, ou seja, a segunda, foram implementadas algumas características não encontradas na primeira versão, tais como:

- Suporte para *enconding* do tipo literal;
- A serialização dos objetos, a qual faz parte de um pacote separado que pode ser utilizado opcionalmente;
- É possível a utilização de um *flag* que indica o recebimento de um pacote .NET para a serialização de objetos, permitindo, assim, a troca de mensagens entre diferentes plataformas.

O objetivo deste pacote é fornecer funcionalidades semelhantes às oferecidas pelos pacotes SOAP para dispositivos móveis, permitindo, nesse sentido, uma utilização mais otimizada dos recursos destes equipamentos.

2.1.9 Universal Description, Discovery and Integration

A *Universal Description, Discovery and Integration* (UDDI) é uma especificação para descrever, descobrir e integrar serviços na Web. Um serviço UDDI é um *web services* que gerencia informações sobre provedores, implementações e metadados de serviços.

A especificação UDDI tem como objetivo disponibilizar um padrão para a publicação e descoberta de serviços. Com a utilização de UDDI, é possível localizar um determinado serviço e obter sua descrição técnica. As informações obtidas no contexto UDDI são classificadas em três categorias principais:

- Páginas Brancas (*White Pages*): Essas incluem informações gerais sobre uma empresa específica, como, por exemplo, nome de um negócio, descrição do negócio, informação de contato, endereço e identificadores conhecidos;
- Páginas Amarelas (*Yellow Pages*): Incluem dados de classificação geral para empresa ou serviço oferecido. Por exemplo, esses dados podem incluir a indústria ou o produto;
- Páginas Verdes (*Green Pages*) – Esta categoria contém as informações técnicas sobre o *web services*. Geralmente, essa informação inclui um apontador (ponteiro) para uma especificação externa e um endereço para invocar o serviço.

Os provedores de serviços utilizam UDDI para publicar os serviços que eles oferecem. Usuários de serviços podem usar UDDI para descobrir serviços que lhes interessem e obter os metadados necessários para utilizar esses serviços, para, logo após, estabalecer uma conexão direta com o *web services* que se está procurando.

2.1.10 BPEL - Business Process Execution Language

Entre os diversos componentes de uma arquitetura orientada a serviços, pode-se destacar como principal os serviços (*web services*). SOA propõe aplicações compostas a partir desses serviços. Com isto, para a criação dessas aplicações, linguagens de composição de serviços surgiram, sendo que a que está tornando-se o padrão é a linguagem BPEL. Seu exemplo pode ser vista na Figura 2.11.

```
         <sequence name="Main">
            <receive name="RecebeParametros" createInstance="yes"
partnerLink="LinkInterface" operation="interfaceCalculadoraOperation"
portType="ns1:interfaceCalculadoraPortType"
variable="InterfaceCalculadoraOperationIn1"/>
            <if name="If">

<condition>contains($InterfaceCalculadoraOperationIn1.body/ns0:operacao,
'soma')</condition>
               <sequence name="Soma">
            <assign name="AtribuiParametrosSoma">
               <copy>

<from>$InterfaceCalculadoraOperationIn1.body/ns0:valor1</from>
                  <to>$SomaIn1.parameters/valor1</to>
               </copy>
               <copy>

<from>$InterfaceCalculadoraOperationIn1.body/ns0:valor2</from>
                  <to>$SomaIn1.parameters/valor2</to>
               </copy>
            </assign>
                  <invoke name="ChamaSoma" partnerLink="LinkCalculadora"
operation="soma" portType="ns2:CalculadoraWS" inputVariable="SomaIn1"
outputVariable="SomaOut1"/>
                  <assign name="AtribuiResultadosSoma">
               <copy>
                  <from>$SomaOut1.parameters/return</from>
                  <to variable="InterfaceCalculadoraOperationOut1"
part="body"/>
               </copy>
            </assign>
               </sequence>
```

Figura 2.11 – Trecho de código BPEL

O projeto da linguagem foi implementado por desenvolvedores da BEA *Systems*, IBM e Microsoft e iniciou-se com o nome BPEL4WS [WEERAWARANA e CURBERA, 2002], em substituição às linguagens IBM WSFL e Microsoft XLANG. Atualmente é conhecido como WS-BPEL ou simplesmente BPEL, sendo que a versão atual, WS-BPEL 2.0, foi especificamente projetada para orquestrar conversações de longa duração entre *web services*.

BPEL é uma linguagem de programação para especificação de processos de negócios que envolvam *web services*. É uma linguagem baseada em XML, projetada para habilitar o compartilhamento de tarefas para um ambiente de computação distribuída ou computação em *grid*, mesmo através de múltiplas organizações, utilizando uma combinação de *web services*.

2.2 Plataformas de Desenvolvimento para Dispositivos Móveis

Devido à grande variedade de dispositivos móveis existentes no mercado e a grande variedade de *hardwares*, a plataforma para o desenvolvimento vai depender do tipo de dispositivo que se quer atender. Neste sentido, este trabalho apresenta três possíveis plataformas para o desenvolvimento: o .Net *Compact Framework*, para dispositivos com sistema operacional Windows Mobile; o Java *Micro Edition*, para aparelhos que suportem a máquina virtual da Sun; e o Android, para dispositivos construídos sob esta arquitetura. A seguir, serão apresentados estes três tipos de plataformas.

2.2.1 .NET *Compact Framework*

Em outubro de 2000, a Microsoft, em conjunto com a Intel e a Hewlett-Packard como co-patrocinadoras, submeteu a ECMA (*European Computer Manufacturer's Association*) um subconjunto do .NET *Framework* com a finalidade de iniciar um estudo para a padronização desta tecnologia da Microsoft [RICHTER, 2005].

O .NET *Framework* é composto por duas partes: o *Common Language Runtime* (CLR) e sua *Framework Class Library* (FCL). O CLR é responsável por interpretar e executar o código-fonte gerado para a arquitetura .NET, pois, em vez das instruções nativas de CPU, este código compilado produz uma linguagem intermediária (*common intermediate language* - CIL). Em tempo de execução, o CLR traduz a CIL em instruções nativas de acordo com a arquitetura e o conjunto de instruções de CPU disponíveis [RICHTER, 2005].

O .NET *Compact Framework* é a plataforma de desenvolvimento para dispositivos móveis da iniciativa Microsoft .NET. Este *framework* é um subconjunto do .NET *Framework*. Segue a Figura 2.12 que demonstra o subconjunto que representa os pacotes de classes disponíveis no .NET *Compact Framework*.

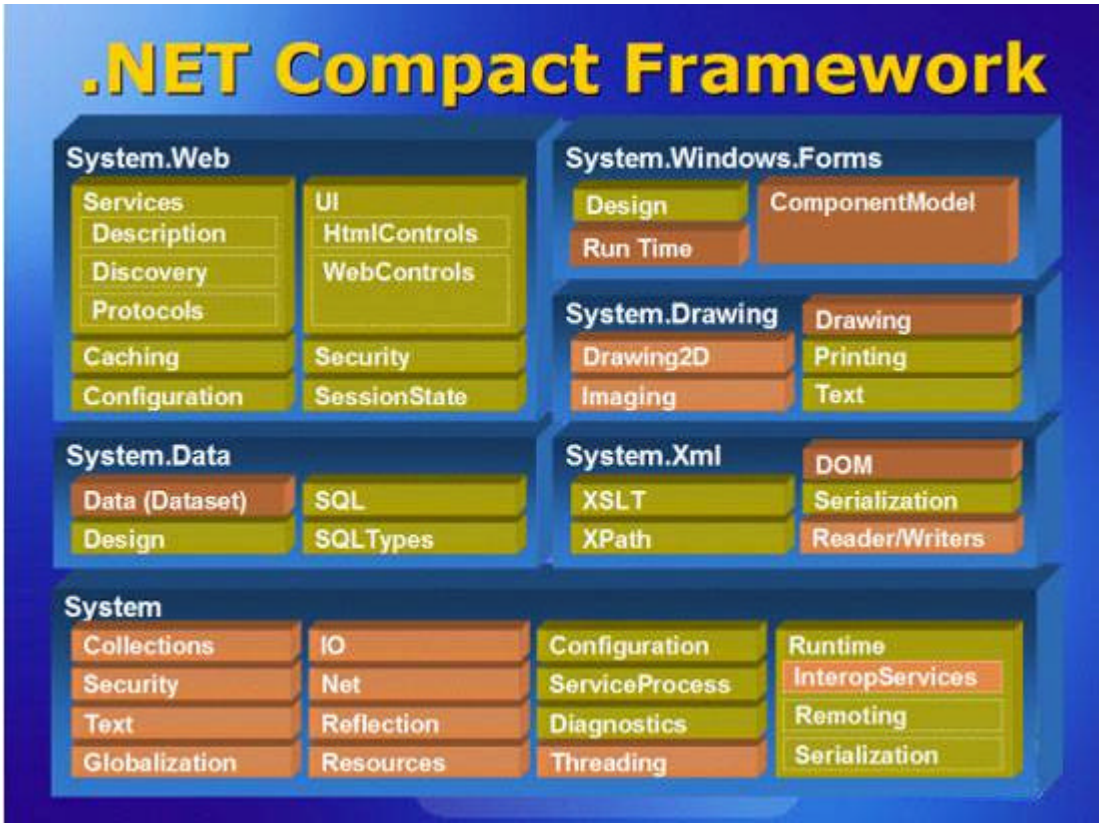

Figura 2.12 – .NET Compact Framework [MSDN CF, 2008]

Juntamente com o *framework*, a Microsoft liberou, a partir da versão 2003 do *Visual Studio*, projetos do tipo *Smart Device Application*, destinado ao desenvolvimento de dispositivos móveis, disponibilizando um emulador para testes em tempo de desenvolvimento, conforme apresenta a Figura 2.13. Por se tratar de um subconjunto do *framework* principal, isso facilitou o desenvolvimento para os desenvolvedores já conhecedores da arquitetura .NET.

Figura 2.13 – Emulador .NET para Dispositivos Móveis

O .NET *Compact Framework*, por ser é um subconjunto do .NET *Framework*, possui muitas das mesmas características chaves encontradas na versão completa do .NET *Framework*, tais como [MSDN CF, 2008]:

- Desenhado para dar base para XML *Web Services*: XML *Web Services* são um modelo de aplicação muito usado para *Smart Devices*, pois seus dispositivos conectados permitem a comunicação com uma gama de outros sistemas. Além disso, sem levar em consideração o sistema operacional ou a linguagem de programação, protocolos padronizados de XML *Web Services* permitem às aplicações se comunicarem com outras;
- Modelo de programação familiar para desenvolvedores *desktop*: O .NET Compact *Framework* usa o mesmo modelo de programação do .NET *Framework*. Este tipo de familiaridade facilita para os desenvolvedores escreverem novas aplicações nesta plataforma;
- *Visual Studio* .NET: É usada a mesma versão do *Visual Studio* .NET para criar aplicações desktop e servidoras que poderão ser executadas no .NET *Compact Framework*. Isso porque o .NET *Compact Framework* utiliza o mesmo modelo de programação e ferramentas para versões *desktop*;

- Segurança: Quanto à segurança, é utilizado o mesmo modelo de segurança do .NET *Framework*;
- Projetado para os recursos limitados dos dispositivos: A plataforma é projetada para detalhar os trabalhos em cima dos recursos limitados até dos pequenos dispositivos, como telefones celulares;
- Alta performance: Aplicações rodando em cima do .NET *Compact Framework* serão executadas em códigos nativos, os quais são produzidos por um compilador *Just-In-Time* (JIT). O uso da tecnologia JIT fornece um grande desempenho na execução do código, tanto que outro dispositivo de sistema programado entenderá como um código interpretado.

2.2.2 Java *Micro Edition*

Java disponibiliza uma plataforma para o desenvolvimento a JEE (Java *Enterprise Edition*). O JEE foi anunciado pela Sun Microsystems em 1999. Esta plataforma é o resultado de esforços da Sun em alinhar as distintas tecnologias Java e APIs em uma plataforma única para desenvolvimento de tipos específicos de aplicações. Atualmente, existem três plataformas diferentes para o desenvolvimento em Java, como apresenta a Figura 2.14 [AHMED e UMRYSH, 2002].

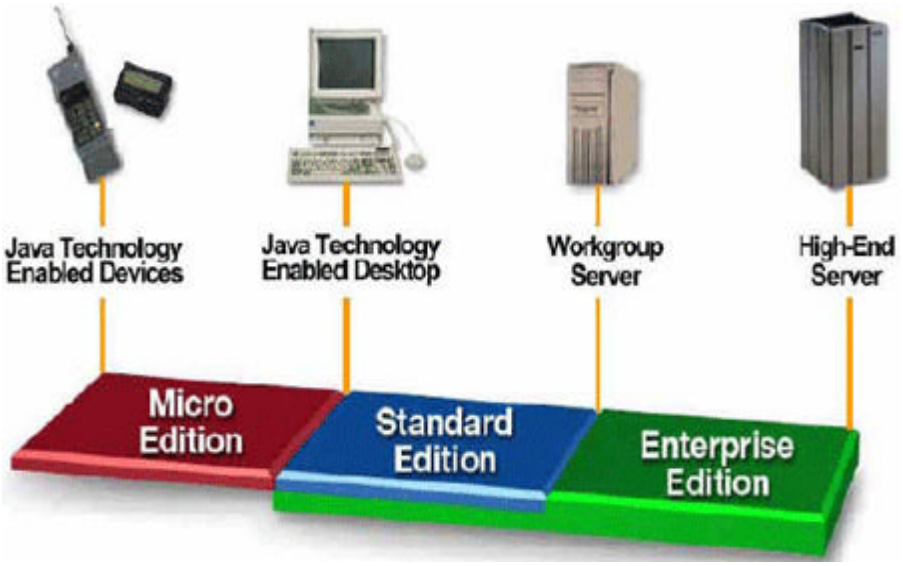

Figura 2.14 – Plataformas Java [JME, 2008]

Cada uma das plataformas pode ser considerada um subconjunto da anterior, conforme descrição abaixo:

- Java *Micro Edition* (JME): A plataforma para o desenvolvimento de *software* para dispositivos móveis, como celulares, pocket PC, etc.;
- Java *Standart Edition* (JSE): Disponibiliza os recursos mais comuns da linguagem Java, como applets, JavaBeans, etc.;
- Java *Enterprise Edition* (JEE): A plataforma para desenvolver aplicações comerciais de maior porte ou para servidores, pois é designada para ser utilizada em conjunto com a JSE.

Como apresentado, Java oferece a plataforma JME para o desenvolvimento de aplicações para dispositivos móveis. JME é uma coleção de tecnologias e especificação para atender aos requisitos dos dispositivos móveis, tais como: celulares, dispositivos embarcados ou dispositivos mais avançados, como *pockets* ou *smartphones*.

A tecnologia JME é baseada em três elementos, como demonstra a Figura 2.15.

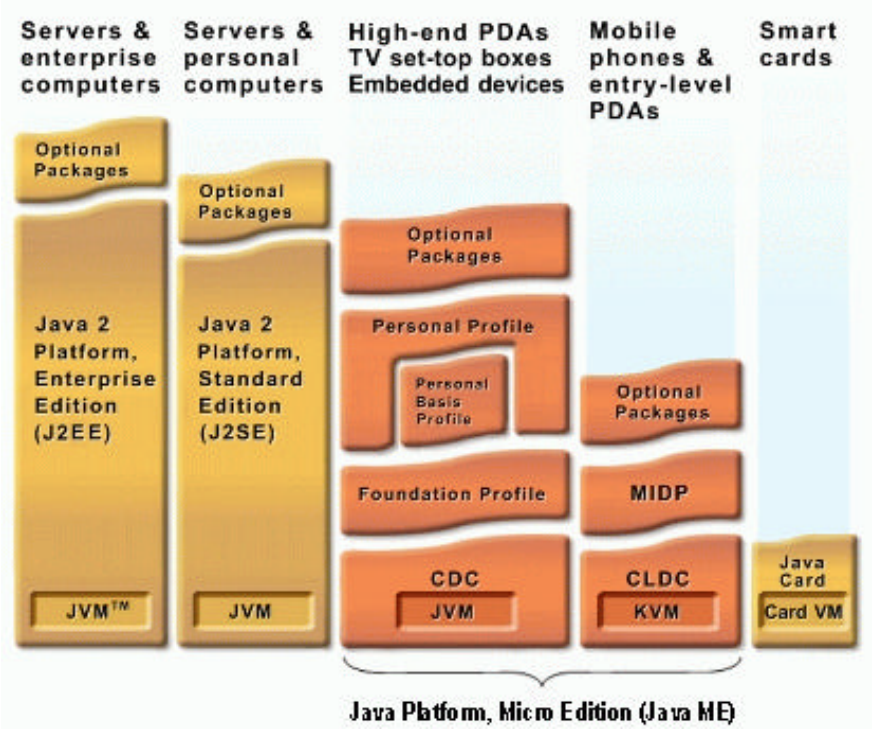

Figura 2.15 – Arquitetura Android [JME, 2008]

Na primeira configuração, são disponibilizados um conjunto básico de bibliotecas e a máquina virtual para dispositivos móveis. Um conjunto de APIs para demais tipos de dispositivos corresponde a uma segunda configuração. Por fim, um pacote adicional de APIs com fins específicos para determinados tipos de dispositivos.

A plataforma JME se divide em duas configurações bases: uma para dispositivos móveis com pequena capacidade de armazenamento e processamento, como telefones celulares ou dispositivos embarcados; e uma segunda configuração para dispositivos com maior poder de processamento, como pockets ou smartphones. A configuração para pequenos dispositivos é chamada de CLDC (*Connected Limited Device Configuration*) e, para dispositivos com maior capacidade, de CDC (*Connected Device Configuration*). As Figuras 2.16 e 2.17 apresentam as duas configurações.

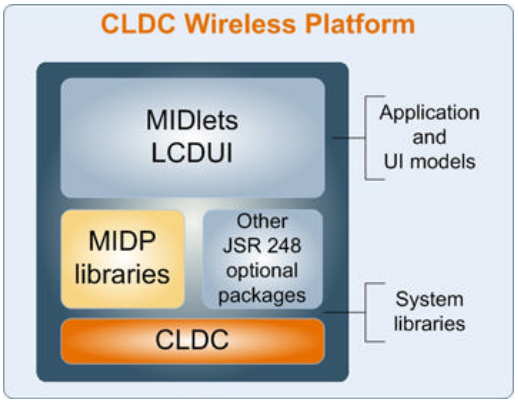

Figura 2.16 – Configuração CLDC [JME, 2008]

Por ser uma plataforma voltada para dispositivos com pouco poder de processamento, possui poucos recursos gráficos e bibliotecas. Suas bibliotecas são disponibilizadas através do MIDP (*Mobile Information Device Profile*), o qual provê um ambiente completo para aplicações de celulares e dispositivos similares.

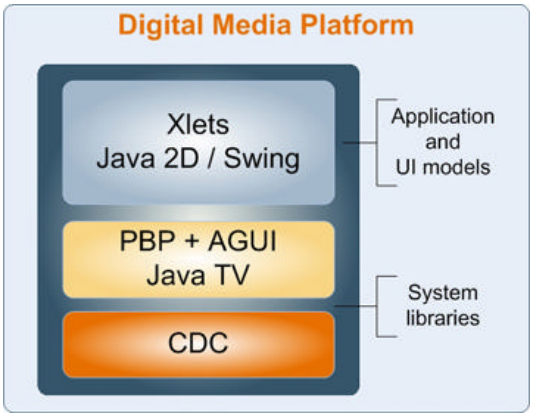

Figura 2.17 – Configuração CDC [JME, 2008]

O objetivo da configuração CDC é de atender dispositivos com maior capacidade com uma tecnologia e ferramentas para desenvolvimento baseada na plataforma Java SE. Com isso, disponibiliza um conjunto maior de bibliotecas, as quais podem atender a dispositivos como PDAs (*Personal Digital Assistent*), smartphones e outros com maior poder de armazenamento e processamento.

A configuração CDC possui três diferentes perfis:

- *The Foundation Profile* (JSR 219);
- *The Personal Basis Profile* (JSR 217);
- *The Personal Profile* (JSR 216).

Para o desenvolvimento, a Sun fornece um emulador que pode simular diferentes tipos de dispositivos móveis, como celulares, smartphones ou PDAs, como mostra a Figura 2.18.

Figura 2.18 – Emulador Java para Dispositivos Móveis

A plataforma JME abrange desde pequenos dispositivos embarcados até dispositivos com maior capacidade. Nesse sentido, o projeto da plataforma permite, de uma forma flexível e eficiente, apoio às necessidades dos serviços de mobilidade. Estes serviços podem ser facilmente portados entre as suas diferentes configurações e perfis, atendendo, assim, aos diferentes canais [JME, 2008].

2.2.3 Android

Foi anunciado pela *Google*, oficialmente em novembro de 2007, o seu sistema operacional voltado para dispositivos móveis, baseado em Linux, chamado de Android. Isso foi o resultado da aquisição, em 2005, de uma empresa de *software* móvel com mesmo nome. Com esta arquitetura, o objetivo é permitir que a companhia faça com que os aplicativos móveis da Google atinjam o maior público possível de usuários de dispositivos móveis.

Apesar do projeto ser amplamente difundido pela *Google*, a responsabilidade do desenvolvimento é da *Open Handset Alliance* [OHA, 2008], um conjunto de 34 empresas do ramo tecnológico e de telecomunicações, tais como: *Telefonica*, *Telecom*, *HTC*, *LG*, *Intel*, *Google*, entre outras.

Diversos recursos estão sendo previstos para os dispositivos que irão utilizar o Android como sistema operacional. Para isso, a plataforma já deve implementar um conjunto de características. As principais características são:

- *Layouts* de dispositivos: Adaptável para *layouts* tradicionais e maiores (*touch screen*);
- Armazenamento: Banco de dados SQLite;
- Conectividade: GSM, CDMA, Bluetooth, EDGE, EV-DO, 3G, Wi-Fi (dependente do hardware);
- Mensagens: SMS e MMS;
- Web browser: Baseado no WebKit *Engine*;
- Máquina virtual: Dalvik *Virtual Machine*, otimizada para dispositivos móveis;
- Mídias suportadas: MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF;
- Suporte a *hardwares* adicionais: Câmera, GPS, bússola, acelerômetro;
- Ambiente rico de desenvolvimento: Emulador de dispositivo, Ferramentas para depuração, Plugin ferramentas de desenvolvimento, tais como Eclipse e Netbeans.

A plataforma Android é composta por um conjunto de ferramentas que abrange, desde sua execução, desenvolvimento e *softwares* específicos. Utilizando a linguagem Java para programação, o *Software Development Kit* (SDK) é o kit de desenvolvimento que disponibiliza as ferramentas e APIs necessários para desenvolver aplicações para esta plataforma [ANDROID, 2008]. A Figura 2.19 exemplifica o conjunto de camadas que compõe esta arquitetura.

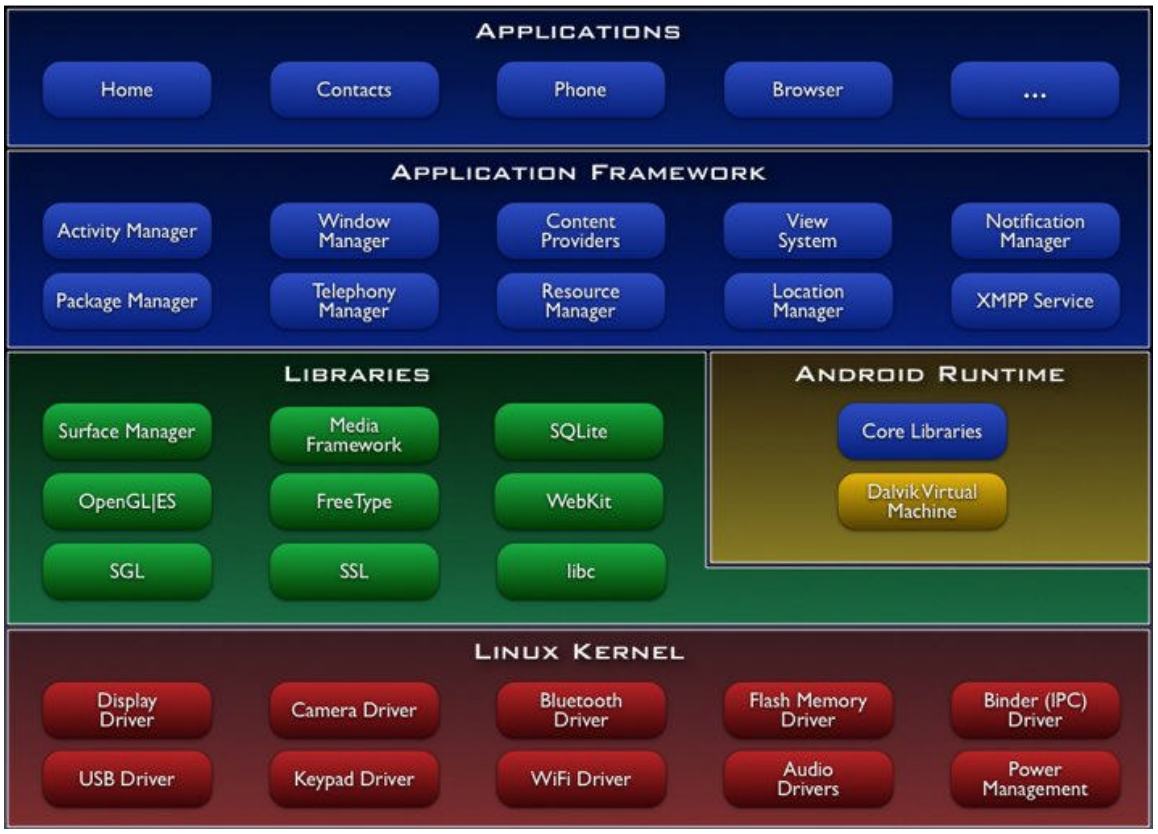

Figura 2.19 – Arquitetura Android [ANDROID, 2008]

A arquitetura Android, conforme mostra a figura, foi dividida em cinco camadas:

1. Linux Kernel: Utiliza o kernel do Linux para os serviços centrais do sistema, tais como segurança, gestão de memória, gestão de processos, armazenamento de arquivos, etc. O kernel também atua como uma camada de abstração entre o *hardware* e o restante do *software*.

2. Bibliotecas: O Android possui um conjunto de bibliotecas desenvolvidas em C/C++ utilizadas por vários componentes do sistema. Estas bibliotecas são expostas para os desenvolvedores através de um *framework*. Abaixo, algumas das principais bibliotecas:

    - System C library: uma implementação derivada da biblioteca C padrão sistema (libc) para dispositivos rodando Linux;

- Media Libraries: baseado no PacketVideo's OpenCORE. As bibliotecas suportam os mais populares formatos de áudio e vídeo, bem como imagens estáticas;
- Surface Manager: fornece acesso ao subsistema de exibição, bem como às múltiplas camadas de aplicações 2D e 3D;
- LibWebCore: um web browser *engine* utilizado tanto no Android Browser quanto para exibições web;
- SGL: o *engine* de gráficos 2D;
- 3D libraries: uma implementação baseada no OpenGL ES 1.0 APIs. As bibliotecas utilizam aceleração 3D via *hardware* (quando disponível) ou o *software* de renderização 3D;
- SQLite: um *engine* de banco de dados relacional disponível para todas as aplicações.

3. Android Runtime: O Android inclui um grupo de bibliotecas que fornece diversas das funcionalidades disponíveis nas principais bibliotecas da linguagem Java para dispositivos móveis (JME).

   Com uma instância da máquina virtual Dalvik, toda aplicação Android é executada em seu próprio processo. O Dalvik foi escrito de forma a executar várias VMs simultâneas. Esta VM executa arquivos com extensão .dex, o qual é otimizado para consumo mínimo de memória. A VM é baseada em registros e roda classes compiladas pela linguagem Java que foram transformadas em arquivos .dex, através da ferramenta "dx" incluída no SDK do Android.

   Para implementar funcionalidades, como o encadeamento e a gestão de baixo nível de memória, o Dalvik VM baseia-se no *kernel* do Linux, como mostra a Figura 2.19.

4. *Framework* de Aplicações: É disponibilizada, para a construção de aplicações na plataforma Android, uma camada de *framework* composta por diversos componentes com o objetivo de simplificar a reutilização de código do

Android. Este mesmo *framework* é utilizado para a construção das aplicações já disponíveis no Android e para as aplicações a serem desenvolvidas. O *framework* é constituído por um conjunto de serviços, incluindo, por exemplo:

- Um grande conjunto de objetos visuais para a construção de aplicações;
- Gerenciador de notificação, que possibilita uma customização para as aplicações de alguns dos recursos visuais para notificação, como alertas, barras de status, entre outros;
- Um gerenciador de atividades, que gerencia todo o ciclo de vida de uma aplicação e a navegação entre essas aplicações;
- Provedor de conteúdo, que habilita as demais aplicações a acessarem dados de outras aplicações, como os contatos do telefone, por exemplo;
- Um gerenciador de recursos, que provê acesso a demais recursos para o desenvolvimento, por exemplo, a localização em mapas, gráficos, etc.

5. Aplicações: Juntamente com o Android, é disponibilizado um conjunto de aplicações, entre elas estão alguns dos exemplos citados abaixo:
   - um cliente de e-mail;
   - programa de SMS;
   - agenda;
   - mapas;
   - navegador;
   - contatos.

Para todos os aplicativos implementados, foi utilizada a linguagem de programação Java.

Além da arquitetura apresentada, a *Open Handset Alliance* desenvolveu um Plugin para o desenvolvimento composto por um ambiente completo para desenvolvimento na

ferramenta Eclipse [ECLIPSE, 2008] com emuladores para testes das aplicações, como apresenta a Figura 2.20.

Figura 2.20 – Emulador para Android

Através destes emuladores, é possível testar as diferentes características dos dispositivos com telas *touch screen* ou com teclado, por exemplo, tornando-se, dessa forma, uma ferramenta essencial para o desenvolvimento.

A próxima seção aborda alguns trabalhos relacionados com o assunto aqui tratado. Trabalhos que possam contribuir para a definição da ferramenta proposta, para execução e composição de serviços em dispositivos móveis, além da análise de impacto nestes serviços.

3 TRABALHOS RELACIONADOS

Neste capítulo, são apresentados quatro trabalhos com assuntos relacionados a esta proposta, uma vez que estes trabalhos contribuíram para elaboração da mesma. O primeiro é um artigo que apresenta um trabalho relacionado à análise de impacto em aplicações orientadas a serviços, enquanto que os próximos dois visam a descrever trabalhos relacionados a QoS. O último artigo, por sua vez, apresenta um *middleware* para a construção de ambientes colaborativos para dispositivos móveis, foco do projeto principal no qual este trabalho está inserido.

3.1 Supporting Change impact Analysis for Service Oriented Business Applications

Com a crescente utilização de serviços para encapsular as funcionalidades dos sistemas, um problema que surge neste cenário, diz respeito à mudança nestes serviços e ao impacto que essa alteração poderá gerar nas demais composições que utilizam o serviço afetado. Neste sentido, Xiao e Zou (2007) propõem em seu artigo uma forma de se estimar os custos de uma alteração nos serviços.

Os autores sugerem que seja efetuada uma análise de impacto em nível de processo de negócio e em nível de código fonte. Para que isso ocorra, as seguintes tarefas devem ser levantadas:

1. Identificação dos componentes que devem ser alterados e armazenados em um *conjunto de componentes de negócios impactados*;
2. Uma análise sobre os dados e dependências de controle dos processos de negócio, gravando no *conjunto de componentes de negócios impactados*;
3. Identificação da ligação entre os componentes de negócio e o código fonte, mapeando todos os componentes;
4. Armazenar todos os códigos fontes identificados na alteração em um *conjunto de códigos impactados*;
5. Uma análise sobre os dados e dependências dos códigos fontes, gravando todas as dependências de código no *conjunto de códigos impactados*;

6. Cálculo de métrica das mudanças, pois a métrica é baseada em toda a propagação das alterações e visa a disponibilizar as informações sobre uma mudança em um processo de negócio.

3.1.1 Análise de Impacto em Processo de Negócio

Para análise em processos de negócio, é definido um *workflow* que consiste em cinco componentes:

- Descrição das tarefas necessárias para se alcançar os objetivos de negócio, uma vez que as tarefas podem ser descritas como um conjunto interno e externo de propriedades;
- Fluxo de controle que determina o caminho para execução das tarefas, pois um conjunto de tarefas pode ser executado em diferentes sequências ou em repetições;
- Fluxo de dados que descreve as entradas e saídas das tarefas;
- Papéis que desempenham as tarefas;
- Recursos necessários para a execução da tarefa.

Os autores identificaram um conjunto básico de alterações que podem ocorrer nos componentes de negócio, descritas no Quadro 3.1.

Quadro 3.1 – Principais Alterações Identificadas

Alteração	**Descrição**	**Impacto nos Componentes de Negócio**
Interna em Propriedades	Modificação interna em uma tarefa sem alteração de interfaces	Modificação de tarefa
Dados de Entrada	Modificação na interface de entrada	Modificação na tarefa e nas tarefas anteriores
Dados de Saída	Modificação na interface de saída	Modificação na tarefa e nas próximas tarefas
Inclusão de Tarefas	Adição de uma nova tarefa com uma interface de entrada e saída	Adicionar tarefa
Exclusão de uma Tarefa	Exclusão de uma tarefa existente com uma interface de entrada e saída	Todas as tarefas que possuem dependência

Com posse das informações sobre os processos que serão afetados por uma determinada alteração, é realizado um mapeamento entre os processos e os códigos-fontes que serão afetados para que seja feita uma análise de impacto sobre o código.

3.1.2 Análise de Impacto em Código-Fonte

Para realizar uma análise de impacto nos códigos, os autores desenvolveram um gráfico de propagação que apresenta todas as entidades envolvidas em uma alteração, como componentes ou métodos. A Figura 3.1, abaixo, apresenta o gráfico citado. A partir deste gráfico, é derivada uma fórmula matemática para medir o esforço necessário para implementar uma mudança de um processo de negócio.

A partir de uma determinada alteração em código, é gerada uma análise sobre a hierarquia das classes afetadas. Procura-se, assim, por todos os relacionamentos de herança com a classe alterada e/ou por todas as classes que executam o método alterado. Estes dados servem para inserir informações no conjunto de códigos impactados, que será a base para o gráfico de propagação de alterações.

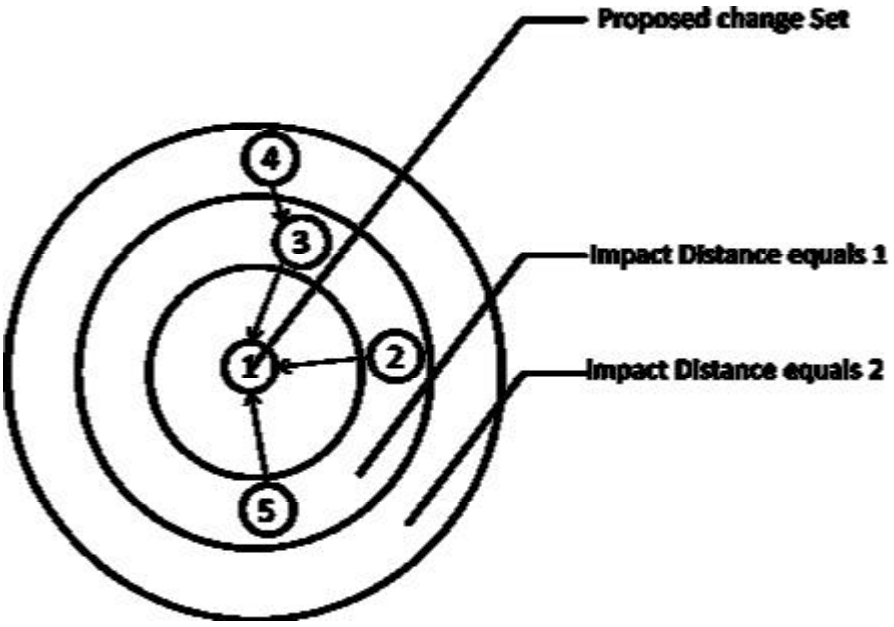

Figura 3.1 – Gráfico de Propagação de Impacto

Na figura apresentada, uma alteração em um método da classe “1” afeta todas as classes que executam este método ou que herdam as características da classe “1”. A partir destes dados, é gerado um gráfico que representa a propagação da alteração, além da distância que cada classe afetada está da classe principal. Por exemplo, o nodo 5 está a uma distância de 1, sendo assim, todos os nodos que estiverem a esta distância estão diretamente relacionados com a alteração. Já o nodo 4 está a uma distância de 2, o que indica que este nodo não foi diretamente afetado, mas possui um grau de interação com o nodo alterado.

3.2 Efficient Algorithms for Web Services Selection with End-to-End QoS Constraints

Neste trabalho, os autores apresentam a construção de uma arquitetura baseada em um *broker* para a seleção de serviços, não deixando de considerar níveis apropriados de QoS (*Quality of Service*), os quais foram estudados pelos autores. Apesar de a proposta de dissertação não ter como foco a seleção de outros serviços em tempo de execução com relação à qualidade destes serviços, este artigo contribui no sentido do entendimento das variáveis que são importantes na hora de mensurar a qualidade dos serviços que fazem parte da composição [ZHANG et al., 2007].

A arquitetura denominada *QBroker* define as seguintes funções: descoberta de serviços, planejamento, seleção de serviços e adaptação. Basicamente, a eficiência do *QBroker* está relacionada ao algoritmo que, no momento em que este precisa ser executado, realiza a seleção do serviço. A Figura 3.2 apresenta uma composição simples de serviço.

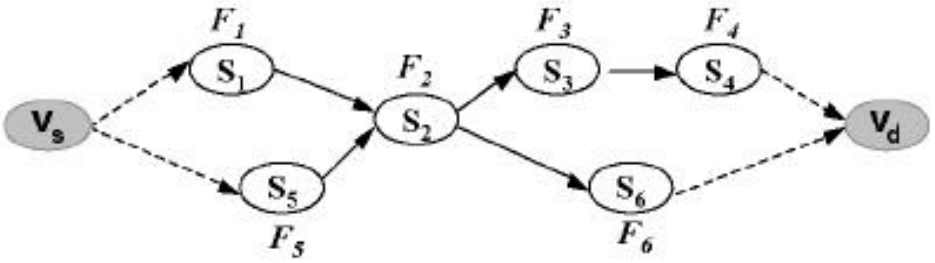

Figura 3.2 – Composição de Serviços

A figura apresenta uma composição com quatro caminhos possíveis para a execução completa do serviço: {*S*1, *S*2, *S*3, *S*4}, {*S*1, *S*2, *S*6}, {*S*5, *S*2, *S*3, *S*4} ou {*S*5, S2, *S*6}. Cada

uma dessas funções Si pode ser executada como um serviço atômico. Os valores de QoS medidos para cada serviço são apresentados na Tabela 3.1.

Tabela 3.1 – Estimativa de Execução dos Serviços

Serviço	Tempo de Resposta	Custo	Disponibilidade
S1	100	50	0,95
S2	195	50	0,98
S3	216	100	0,94
S4	150	60	0,93
S5	231	150	0,96
S6	162	100	0,97

As principais variáveis medidas quanto ao QoS são o tempo de resposta total do serviço, o custo de execução e o percentual de disponibilidade do serviço. Uma possível avaliação do melhor caminho seria realizar a soma de todos os serviços que fazem parte de um determinado caminho e avaliar o que terá o menor custo de tempo e o que manterá o maior percentual de disponibilidade. Contudo, estes valores podem variar em tempo de execução, pois dependem de diversos fatores externos, como velocidade de conexão, por exemplo. Tornando, assim, inviável executar todas estas estimativas em cada execução de serviço.

Com isso, os autores propõem a utilização de algoritmos que visam a minimizar a execução de composições selecionando os melhores caminhos. A proposta considera seis estruturas possíveis para composição, como mostra a Figura 3.3.

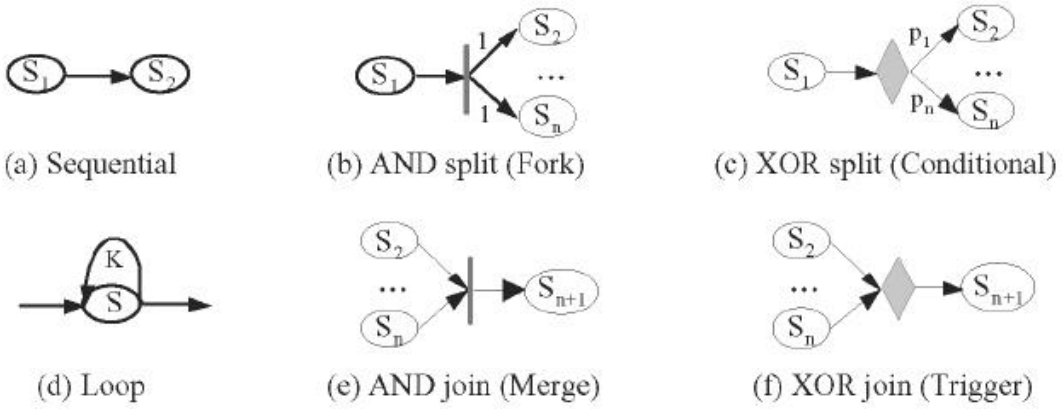

Figura 3.3 – Estrutura de Condições

A partir de uma estrutura básica, é realizada uma composição, como apresenta a Figura 3.4, na qual são aplicadas as estimativas com os algoritmos propostos para avaliação do melhor caminho para a execução.

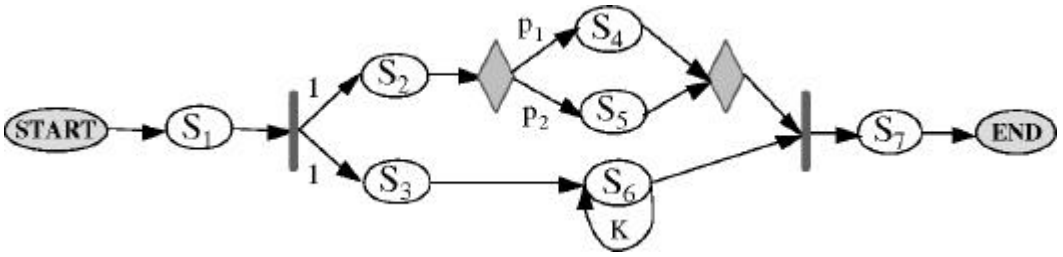

Figura 3.4 – Composição com Seleção de Serviços

Para o cálculo do melhor caminho, são considerados dois fatores principais: tempo de execução (T) e custo (C), os quais são medidos conforme fórmula abaixo:

$$T = t1 + \text{Max}(t2 + p1 * t4 + p2 * t5,\ t3 + K * t6) + t7$$

$$C = c1 + c2 + p1 * c4 + p2 * c5 + c3 + K * c6 + c7$$

Onde *t* representa o tempo de resposta do serviço, *c* o custo individual dos serviços, *p* a probabilidade de execução e K o total de vezes que determinado serviço é executado (loop). A partir destes dados, os algoritmos foram submetidos à avaliação para determinar a eficiência da arquitetura *QBroker*.

3.3 Toward Semantic QoS aware Web Services: Issues, Related Studies and Experience

Neste artigo, é apresentada uma visão geral quanto à utilização dos conceitos de QoS em *web services*. Os autores relacionaram os principais tópicos, tais como arquitetura de QoS para *web services*, classificação de QoS, ontologias voltadas a QoS e linguagens para especificação. Além disso, eles direcionaram seus estudos a trabalhos relacionados na área, fornecendo, assim, uma base de conhecimentos mínimos para se iniciar um estudo sobre QoS em *web services* [ZHOU e NIEMELA, 2006].

A qualidade, como foi apresentada pelos autores em um sistema, componente ou processo, representa o grau de adequação deste *software* a determinados requisitos ou necessidades do usuário. O artigo se refere às propriedades não funcionais dos serviços,

como, por exemplo, o nível de aceitação do serviço, que corresponde ao "contrato" entre o provedor de serviços e o consumidor que deve ter o serviço garantido por este provedor. A seguir, são detalhados os principais tópicos relacionados no artigo quanto à utilização de QoS em *web services*.

3.3.1 Arquitetura QoS para *Web Services*

O Quadro 3.2 apresenta as principais arquiteturas de QoS para *web services* estudadas pelos autores. Aqui, duas das arquiteturas são baseadas em SLA (*Service Level Agreement*), que corresponde ao grau de aceitação entre o provedor de serviços e o consumidor. Já as demais arquiteturas correspondem à integração dos serviços ou descoberta.

Quadro 3.2 – Arquiteturas de QoS para Web Services

Modelo	**Objetivo**	**Técnica Utilizada**
WSLA Framework	Definição e monitoração de SLAs para web services	Linguagem WSLA
BMP	Gerenciamento de SLAs entre dois serviços	Linguagem WSML
Modelo Baseado em SOA	Descoberta de serviços aplicando QoS	Estende UDDI
Modelo WS QoS	QoS de integração dos serviços	Qos XML schema
QuA	QoS aplicado a objetos distribuídos	Middleware

Um modelo interessante apresentado que se utiliza apenas dos recursos atuais, apenas estendendo sua utilização, é o baseado em SOA. Dentro dele, é apresentado um novo modelo que pode coexistir com os atuais registros UDDI, estendendo este para sua utilização, acrescentando informações referentes à qualidade dos serviços, à estrutura atual de UDDI.

3.3.2 Ontologias

O artigo apresenta cinco ontologias que serviram ao estudo. O objetivo destas ontologias é de definir um vocabulário comum, métricas e até mesmo propriedades que sejam comuns às diferentes áreas relacionadas à QoS. Algumas ontologias, como a FIPA QoS, por exemplo, definem uma ontologia referente à comunicação para a utilização de QoS, enquanto que outras, como a WS QoS Ontologia, definem um suporte para a descoberta de serviços utilizando-se de QoS.

3.3.3 Linguagens para Especificação

Todas as linguagens apresentadas pelos autores, tais como *Web Services Management Language* (WSML), *Web Services Level Agreement* (WSAL) e *Web Services Offer Language* (WSOL) são baseadas na linguagem XML. Estas linguagens visam descrever um modelo comum na utilização de QoS para *web services*.

3.4 Middleware for Multimedia Mobile Collaborative System

Os autores apresentam, neste artigo, um *middleware* que objetiva a facilitar a construção de sistemas colaborativos para dispositivos móveis focados em aplicações multimídia. Neste trabalho, é apresentada a arquitetura básica de um *framework* para auxílio na criação de sistemas colaborativos ubíquos. Inicialmente, é exposto um *framework* conceitual para que suporte a criação de aplicações colaborativas para dispositivos móveis. Seguindo, a Figura 3.5 apresenta as camadas desenvolvidas para o *framework* [SU et al., 2004].

Figura 3.5 - *Framework* do Ambiente Colaborativo Móvel

Conforme mostra a figura, o *framework* foi dividido em quatro componentes principais:

- Camada de Geração de Conteúdo: nesta camada, o servidor de conteúdo gera o conteúdo baseado em uma requisição do cliente. A mensagem de requisição do cliente consiste no perfil do dispositivo, estado prévio da rede, e na URL solicitada. Assim, se os dados estiverem disponíveis, o servidor verifica a disponibilidade dos dados e o conteúdo será gerado e entregue ao cliente, baseado no tipo do dispositivo que foi solicitado e também na conexão de rede;
- Camada de Comunicação: mantém em cada sessão a interação e entrega de mensagens entre o cliente e o servidor. Esta camada é responsável por descobrir o estado da rede e decidir se armazena, para seu posterior envio as mensagens em filas, para o caso de não existir conectividade da rede;
- Camada de Consumo e Re-geração de Conteúdo: nesta camada, as mensagens são combinadas e seus conteúdos enviados para a camada de visualização para exibição. Caso o cliente deseje fazer qualquer modificação no conteúdo, este será re-gerado. Assim, o conteúdo modificado pode ser enviado a outro usuário diretamente ou ao servidor;
- Camada de Visualização de Conteúdo: uma vez que o conteúdo chegue à camada de visualização, este objeto é transformado em uma estrutura DOM (*Document Object Model*), para, então, ter a utilização de um visualizador para apresentar o conteúdo disponibilizado.

De uma forma resumida, estas quatro camadas apresentadas fornecem suporte para busca e apresentação de conteúdo para ambientes heterogêneos. Ao se considerar a interação permitida aos usuários em páginas WEB, isso possibilita a colaboração e execução de sistemas complexos em dispositivos móveis, possibilitando a comunicação entre diferentes plataformas. A Figura 3.6 apresenta a arquitetura para um sistema colaborativo em dispositivos móveis construído a partir *framework* proposto.

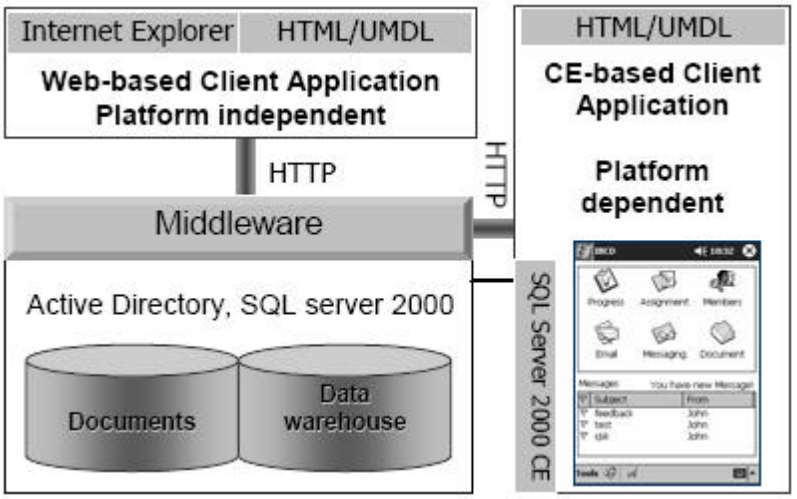

Figura 3.6 - Arquitetura de um Sistema Colaborativo Móvel

No protótipo desenvolvido, disponibilizando diferentes sistemas para os usuários a partir do *middleware* desenvolvido, o sistema permite a comunicação entre os membros de um grupo através de e-mail, chat ou mensagens de voz, além do compartilhamento de arquivos. Desta forma, é apresentada a utilização da arquitetura proposta para a construção de sistemas colaborativos para dispositivos móveis.

No capítulo seguinte, será apresentada a modelagem da ferramenta desenvolvida. Procura-se mostrar o desenvolvimento dos requisitos necessários e de como a ferramenta permitirá análises sobre as composições de serviços.

4 GIMPACT: ANÁLISE DE IMPACTO NA COMPOSIÇÃO DE SERVIÇOS

Este capítulo apresenta o projeto para construção de uma ferramenta que visa dar suporte para a análise de impacto em uma arquitetura SOA para dispositivos móveis. Esta ferramenta está inserida em um projeto maior que tem como objetivo disponibilizar uma meta-arquitetura para dar suporte a ambientes colaborativos para dispositivos móveis.

Como trabalho de dissertação, esta pesquisa será focada nas novas tendências de uso de dispositivos móveis na formação de ambientes colaborativos que maximizem o uso deste tipo de dispositivo.

4.1 Projeto de Pesquisa U-SOA

O projeto da ferramenta gImpact faz parte de um projeto maior de nome U-SOA (*Ubiquitous SOA Framework*) [ZANUZ et al., 2008], de desenvolvimento de um grupo de pesquisa da universidade. U-SOA pretende ser um *framework* para a construção de sistemas colaborativos ubíquos baseados em arquiteturas orientadas a serviços. U-SOA prevê também estruturas capazes de fornecerem mapas das composições de serviços e dos respectivos serviços utilizados nessas composições, permitindo, dessa forma, que diversas análises de impacto e de QoS possam ser construídas a partir desses mapas, foco deste trabalho em questão.

U-SOA, conforme ilustra a figura 4.1, pode ser vista como um conjunto de tecnologias separadas em camadas. As camadas envolvidas na arquitetura contemplam desde níveis mais baixos, no qual se encontra o sistema operacional Android, até o topo da pilha, onde se encontram as aplicações colaborativas ubíquas suportadas pela arquitetura.

Atualmente, os seguintes trabalhos já foram ou estão sendo desenvolvidos:

1. Servidor de *web services* SOAP Server;
2. Motor de execução SmallSOA;
3. Linguagem de composição de serviços inSOA, anteriormente chamada de SQLSOA [ZANUZ et al., 2008];
4. Ferramenta para análise de impacto na composição de serviços gImpact;

5. Robô para descobrimento de serviços WSBot;

6. Interface para composição de serviços em dispositivos móveis.

Com exceção do primeiro item, os demais são de desenvolvimento do grupo de pesquisa citado.

No topo da arquitetura são encontradas as aplicações colaborativas ubíquas que poderão ser criadas a partir do suporte dado pelos componentes desenvolvidos nas camadas inferiores de U-SOA.

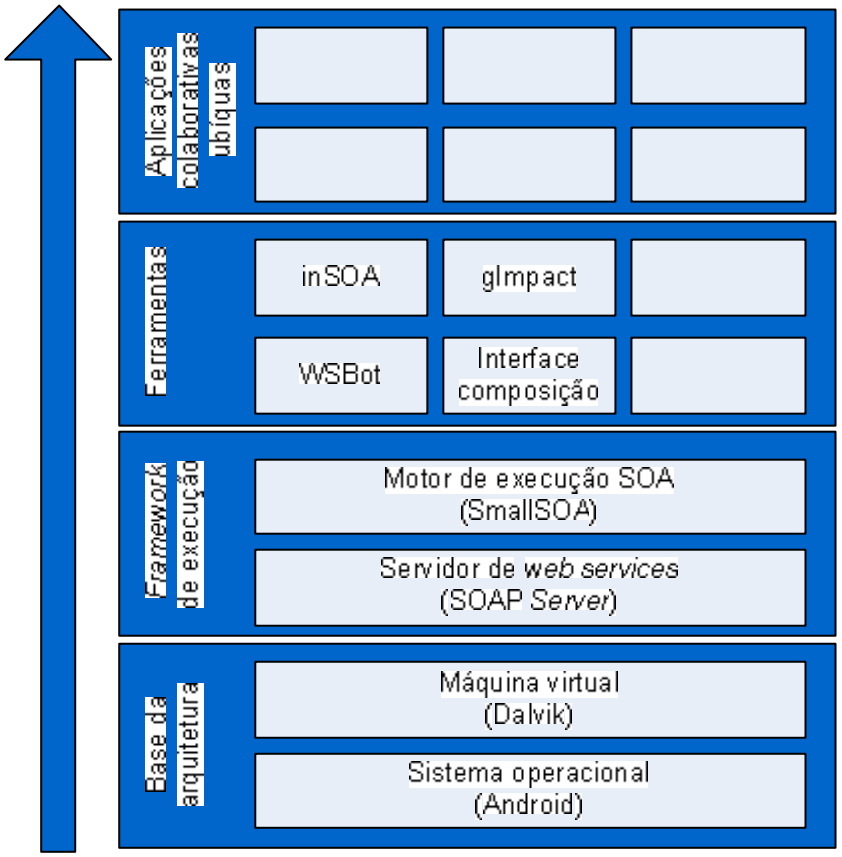

Figura 4.1 – Arquitetura U-SOA

Esta arquitetura objetiva disponibilizar a composição e a execução de serviços em dispositivos móveis para sua utilização em aplicações colaborativas ubíquas. Para isso, é necessário um servidor de *Web Services* sob uma plataforma para dispositivos móveis, para o trabalho por se tratar de uma plataforma de *software* livre se optou pelo Android [ANDROID, 2008], como demonstrado na figura. Acima desta camada, encontra-se a camada de execução de serviços e composições. Esta camada não faz parte do escopo deste trabalho, porém será

utilizada na execução dos serviços. Nesse sentido, a ferramenta para análise de impacto, proposta deste trabalho, está localizado acima da camada de execução de serviços, provendo, assim, um conjunto de funcionalidades para análise das composições.

4.2 Infra-Estrutura para Execução e Composição de Serviços

Como base para a camada de infra-estrutura, este trabalho tem como finalidade implementar uma ferramenta que disponibilize um conjunto de transações para análise de impacto nos serviços ou composições disponibilizados em um determinado servidor ou em todos os servidores que utilizem a arquitetura proposta. Atualmente, a arquitetura para publicação de serviços ou composições não permite que seja analisada a árvore de composições, uma vez que a pesquisa é realizada apenas sobre o arquivo WSDL, o qual descreve apenas os parâmetros de entrada e saída de um serviço e não sua composição. Para que seja possível a realização de uma análise em composições, é proposta a criação de uma nova arquitetura para a disponibilização de serviços e composições, conforme apresenta a Figura 4.2.

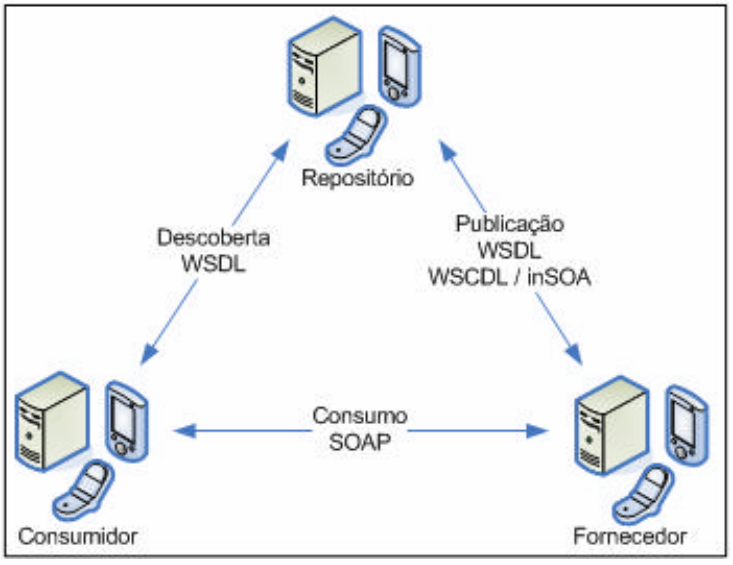

Figura 4.2 – Modelo Proposto de *Web Services* e Composição

Neste novo modelo, um dispositivo móvel, ou uma estação cliente, pode assumir qualquer um dos três papéis possíveis na execução de um serviço, ou seja, o papel do cliente, do provedor de serviços ou do servidor de publicação. Cabe dizer que a principal diferença do

modelo atual de execução de serviços está na publicação. Assim, é utilizada, nesta etapa, uma linguagem que descreve a composição dos serviços, inSOA [ZANUZ et al., 2008]. Através da representação desta linguagem, é possível realizar uma leitura na árvore de execução e apresentar todos os serviços que fazem parte de uma composição. Um exemplo da publicação das composições conforme novo modelo é apresentada na Figura 4.3.

```
- <inSOA id="airportComposition" tags="travel, money" hash="67dd30dfa352b3ea329e2fbb762bf034abb5e50e">
    <inSOAScript/>
  - <Invokes>
    - <Invoke Method="SOAP" Namespace="http://www.webserviceX.NET"
      EndPoint="http://www.webserviceX.NET/airport.asmx" Operation="getAirportInformationByAirportCode"
      Alias="a" Pipe="/Envelope/Body/getAirportInformationByAirportCodeResponse
      /getAirportInformationByAirportCodeResult" Into="airport" ParallelismLevel="1">
      - <Parameters>
          <Parameter Name="airportCode" Value="POA" Type=""/>
        </Parameters>
      </Invoke>
    </Invokes>
    <Wheres> </Wheres>
  - <Returns>
      <Return ReturnExpression="airport" TypeReturnExpression=""/>
    </Returns>
    <Fails> </Fails>
  </inSOA>
```

Figura 4.3 – XML para Representação de Serviços na Linguagem inSOA

Para interpretar as composições, todas as camadas fazem uso de uma linguagem de composição comum (inSOA). Desta forma, é possível a comunicação entre as diversas camadas da arquitetura e a leitura da árvore de execução de uma composição, já que o arquivo WSDL não permite a visualização das composições realizadas, pois esta visualização só se torna visível em nível de programação com a linguagem BPEL (*Business Process Execution Language*). Esta linguagem, BPEL, implementa composição e orquestração de serviços a partir de um modelo baseado na linguagem XML, porém, quando o serviço é exposto, apenas o arquivo WSDL é encaminhado, não permitindo o conhecimento de quais serviços estão agregando determinada composição. Por este motivo, faz-se necessária a utilização de uma linguagem que exponha a composição realizada. A linguagem inSOA, através da *tag "Invoke"* expõe os serviços que fazem parte da composição, assim a partir do primeiro nível exposto de serviços é possível a pesquisa do restante da árvore.

4.3 Ferramenta para Análise de Impacto

Uma questão importante levantada por Metsker (2004), em seu livro, diz respeito a alterações em funcionalidades já implementadas ou à sua disponibilidade. Quando se faz indispensável a utilização de um serviço, o cliente busca as informações necessárias deste serviço e o executa. No entanto, durante a composição dos serviços, surge um problema. Como identificar a melhor composição para o atendimento do negócio?

Em um âmbito empresarial, onde sistemas de grande porte são desenvolvidos e a integração entre estes sistemas ocorre com muita frequência, as informações sobre o impacto na utilização dos serviços podem reduzir substancialmente o custo em sua execução. Uma vez que, ao se realizar a análise de impacto pode-se detectar a melhor composição para a necessidade do negócio, visando o melhor desempenho do serviço.

Outra questão importante quando se fala na execução de serviços diz respeito à sua disponibilidade. Por se tratar de serviços disponibilizados em uma rede, estes podem ser removidos sem que todos os clientes tenham consciência desta alteração, ocorrendo, assim, um erro no momento de sua execução, ou ainda podendo existir o mesmo serviço em servidores diferentes, sendo que um, por motivos de infra-estrutura, poderá possuir uma maior disponibilidade que outro. O termo qualidade de serviço, também conhecido como QoS (*Quality of Service*), trata da disponibilidade dos serviços, sejam estes relacionados a diferentes áreas, como telecomunicações, redes de computadores ou outras áreas.

Nestas duas questões é que se concentra a construção de uma ferramenta para a análise de serviços em um ambiente descentralizado, como a Internet.

4.3.1 Análise de Impacto

Determinar o impacto ou custo de execução de um serviço não é uma tarefa trivial, pois envolve diversas variáveis de ambiente e complexidade da composição. Neste primeiro cenário, o trabalho visa a fornecer um conjunto de funcionalidades que permitam ao usuário efetuar uma análise sobre o impacto em serviços. Para isso, é necessário que seja construída a árvore de composição através do conhecimento de uma linguagem específica de composição para realização da análise.

Tendo em vista ferramentas disponibilizadas no mercado para a composição de serviços, como NetBeans [NETBEANS, 2008], JDeveloper [JDEV, 2008] ou Eclipse

[ECLIPSE, 2008], estas permitem apenas que, no caso de serviços de BPEL, seja realizada uma análise ou *refactoring* sobre a composição em serviços que estejam em formato de código fonte. Neste sentido, a ferramenta proposta para a análise se diferencia das demais no ponto que efetua a análise do impacto no arquivo que expõe a publicação dos serviços e não no código fonte. Desta forma, é possível realizar a análise em serviços já compilados ou em servidores diferentes, ou até mesmo serviços que estejam publicados na Internet, não importando o servidor, desde que os servidores tenham instalada a infra-estrutura necessária para a publicação de serviços proposta. O Quadro 4.1 apresenta um comparativo entre as ferramentas citadas e a aplicação proposta neste trabalho.

Já o conceito de *refactoring* está atrelado às mudanças feitas na estrutura interna do *software* para torná-lo mais fácil de entender e com menores custos para realizar modificações, sendo que seu comportamento deve ser mantido inalterado [GARCIA et al., 2004]. O trabalho tem como objetivo permitir, através da análise de impacto estimar a complexidade de composições, para que sejam realizadas otimizações nos serviços, permitindo que o usuário selecione os melhores serviços ou que sejam realizados *refactoring* nos serviços já implantados.

Quadro 4.1 – *Refactoring* em Serviços

***Refactoring* em**	**NetBeans**	**Eclipse**	**JDeveloper**	**gImpact**
Serviços no mesmo Projeto	Sim	Sim	Sim	Sim
Serviços em outros Projetos	Sim	Sim	Sim	Sim
Serviços Compilados	Não	Não	Não	Sim
Serviços na Internet	Não	Não	Não	Sim

Para a análise em composições já implantadas, é necessário consultar, através de uma linguagem específica a árvore de composição dos serviços. Um exemplo dessa árvore pode ser visto na Figura 4.4, na qual se está pesquisando, por exemplo, o impacto em um serviço chamado “S1”.

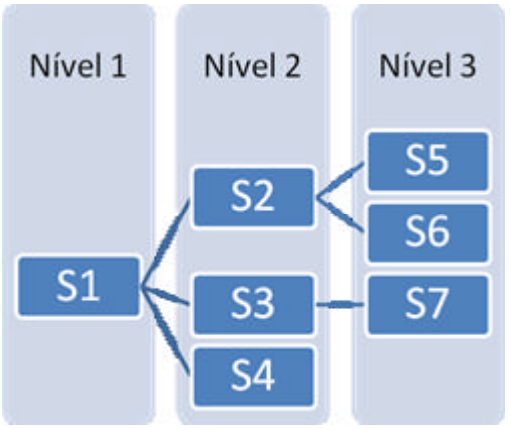

Figura 4.4 – Árvore de Composição de Serviços

Neste exemplo, a árvore retornaria a composição do serviço “S1”. Neste serviço, além da árvore de composição, deve ser retornado em qual nível de profundidade se encontram os serviços. A partir da árvore gerada, o objetivo é fornecer as informações referentes ao impacto no serviço de acordo com a complexidade em suas composições, permitindo assim, uma otimização nas composições.

A ferramenta identifica possíveis erros na composição, como a chamada recursiva de serviços ou um encadeamento entre eles que resulte em um erro, como apresentado na Figura 4.5. Nesta figura, nota-se que a análise do impacto sobre o serviço “S1” retorna sua árvore de composições, e, por fim, o serviço “S5” volta a executar o “S1” em sua composição, apresentando possivelmente um erro na composição de “S1” ou “S5”.

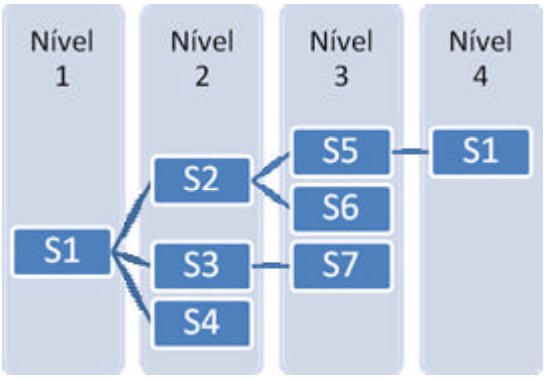

Figura 4.5 – Árvore Representando um Serviço Recursivo

O fato de ser permitido que os serviços estejam fisicamente separados em diversos servidores ou dispositivos móveis deve ser considerado na montagem da árvore de execução.

Uma vez que a análise de um determinado serviço pode não retornar a árvore completa, pois um dispositivo que continha uma composição pode estar desligado ou inacessível naquele momento. Por exemplo, considerando que o serviço “S2”, apresentado na figura 4.5, esteja em um celular que se encontra desligado no momento da análise, a árvore seria modificada conforme apresenta a Figura 4.6. Assim, a ferramenta deve sinalizar que a montagem da árvore foi interrompida em um determinado ponto.

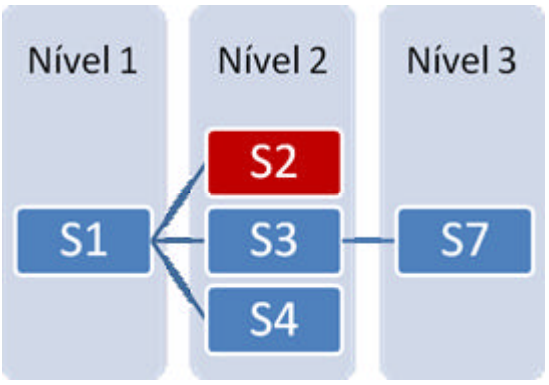

Figura 4.6 – Representação de um Serviço Interrompido

Xiao e Zou (2007) propõem, em seu trabalho, uma fórmula para quantificar o custo de alterações em componentes. Nesta fórmula, os autores consideram todos os métodos afetados por uma alteração e a distância que os métodos afetados se encontram do componente alterado, abaixo a fórmula proposta:

$$I(M) = \sum_{M_j \in S} \left(\frac{1}{D(M_j)} \right)$$

A fórmula apresentada quantifica o custo de uma alteração realizando um somatório dos métodos afetados (M) multiplicados pela distância (D) na qual ele se encontra do método principal. Quanto maior a distância entre os métodos, menor é o impacto estimado para aquele serviço.

Este trabalho serviu como base inicial para a fórmula do presente trabalho que visa quantificar o impacto ou complexidade de uma composição. Para quantificar a análise de impacto, além dos fatores apresentados na fórmula original, como a distância em que um serviço se encontra na árvore de execução, serão considerados outros fatores, tais como a

qualidade de conexão, a localidade do serviço, se os serviços afetados encontram-se no mesmo servidor ou em outros servidores, uma vez que os serviços podem estar hospedados em dispositivos móveis. Abaixo é apresentada a fórmula proposta:

$$I(S) = \sum_{S_i \in S} (D(S_i) + L(S_i) + ((1 - Q(S_i)) * P))$$

Desta forma, a proposta se diferencia do modelo estudado pelo fato de que a medida que a distância aumenta o impacto deve ser maior, pois o impacto se refere a complexidade da composição, diferente da fórmula original onde quanto maior a distância o impacto da alteração é menor. E por acrescentar fatores de qualidade dos serviços, além da localização física. Abaixo a descrição de cada variável:

- *I:* corresponde ao impacto;
- *S:* aos serviços;
- *D:* distância que o serviço se encontra na árvore da composição inicial;
- *L:* localização do serviço, deve ser 0 caso se encontre no mesmo servidor ou 1 se estiver localizado em outro servidor, acrescentando assim uma constante no resultado final calculado;
- *Q:* indicando o percentual da qualidade da conexão com o serviço;
- *P:* indica o percentual de importância que possui a qualidade para o usuário, este parâmetro é configurado na ferramenta.

4.3.2 Análise de Disponibilidade

Ao analisar aspectos, como a qualidade da conexão, a localização dos serviços, a proposta refere-se às propriedades não funcionais dos serviços. Segundo Koscianski e Soares [KOSCIANSKI e SOARES, 2006] "a disponibilidade diz respeito à fração de tempo durante o qual um produto pode ser utilizado.".

Um aspecto importante a ser considerado na utilização de serviços diz respeito à disponibilidade, por se tratarem de *softwares* que se encontram distribuídos na Internet. Uma das características de serviços disponibilizados na Internet é a volatilidade em sua utilização.

Quando há congestionamento na rede, pacotes de dados podem ser descartados sem distinção. Assim, não há garantia de que o serviço será realizado com sucesso, nem mesmo do desempenho na execução deste serviço. Desta forma, uma composição de serviços construída pode não funcionar em um momento, pois um dos serviços que compõe o projeto pode ter sido removido, alterado ou apenas estar indisponível naquele momento.

Nesta abordagem, o trabalho tem como objetivo disponibilizar funcionalidades que executem uma análise de disponibilidade de uma determinada composição. Esta funcionalidade visa identificar e testar a qualidade da conexão com os serviços necessários.

Para a obtenção de dados quanto a disponibilidade de um determinado serviço, é necessário que estas informações sejam armazenadas. Para isto foi criada uma base de dados local, para cada celular, assim cada vez que um serviço é pesquisado esta informação é armazenada, tanto para sucesso quanto para falha. Desta forma é possível a obtenção do histórico de disponibilidade do serviço, a Figura 4.7 exemplifica a arquitetura utilizada para o armazenamento.

Figura 4.7 – Armazenamento para Cálculo de Disponibilidade

Nesse sentido, esta análise fornece informações a respeito da qualidade dos serviços, permitindo identificar quando um determinado serviço possui uma maior ou menor disponibilidade para sua execução, permitindo que esta informação seja avaliada para tomada de decisões por parte do usuário.

4.3.3 Arquitetura para o Desenvolvimento

Por se tratar de uma plataforma aberta e por possuir como base uma linguagem de programação amplamente difundida, o Java, a arquitetura selecionada para o desenvolvimento

do projeto foi o Android [ANDROID, 2008]. Além dos itens mencionados, o Android possui um extenso conjunto de bibliotecas desenvolvidas para a plataforma, além das bibliotecas Java já disponibilizadas para a plataforma Java *Micro Edition*.

A plataforma Android é composta por um conjunto de ferramentas que abrange, desde sua execução, desenvolvimento e *softwares* específicos, assim dando autonomia para todo o desenvolvimento do trabalho proposto.

4.4 Modelagem gImpact

Esta seção apresenta a modelagem da ferramenta gImpact. Será apresentada uma breve descrição dos objetivos da aplicação, os requisitos a serem implementados e o projeto modelado para o desenvolvimento da ferramenta. O modelo permite especificar a estrutura de um sistema ajudando a visualizar como este foi desenvolvido [BOOCH et. al., 2005].

Para desenvolvimento optou-se pela arquitetura voltada a dispositivos móveis, pelo fato deste trabalho estar inserido em um projeto cujo objetivo é disponibilizar uma arquitetura ubíqua colaborativa. Nas próximas seções serão descritas em detalhes a modelagem através da representação em UML (*Unified Modeling Language*) [FOWLER e SCOTT, 2000] para o desenvolvimento da ferramenta.

4.4.1 Requisitos

Requisitos são capacidades e condições às quais o sistema deve atender [JACOBSON et al., 1999]. Para que a ferramenta gImpact seja construída como um componente a ser utilizado na arquitetura U-SOA, requisitos funcionais e não-funcionais precisam ser atendidos. Seguindo o paradigma da orientação a objetos, no qual este trabalho esta inserido, a forma mais comum para representar requisitos funcionais é através de casos de uso [BOOCH et. al., 2005]. Requisitos não-funcionais, por sua vez, normalmente são descritos agrupados por categorias, tais como usabilidade, desempenho e segurança [LARMAN, 2004].

A ferramenta implementada possui as funcionalidades para que um usuário possa efetuar uma análise de impacto sobre serviços implantados e consultas sobre esta análise. Para isto, foram definidos requisitos principais que deveriam ser atendidos: visualização da árvore de composições, uma análise de impacto das composições e um histórico de sua execução. A

partir dos requisitos principais surgem requisitos secundários, são eles: consulta de estatísticas das composições e configurações do sistema. A partir da identificação dos requisitos foi realizada a modelagem utilizado-se o diagrama de Casos de Uso. A Figura 4.8 apresenta os Casos de Uso, expressando os requisitos necessários da ferramenta implementada.

Durante a modelagem foram identificados três atores envolvidos:

- *User*: o próprio usuário do sistema;
- *SOA Engine*: o motor de execução que deverá estar instalado nos dispositivos em questão;
- *Device*: todos os dispositivos onde se encontram as composições, podendo estes serem móveis ou não.

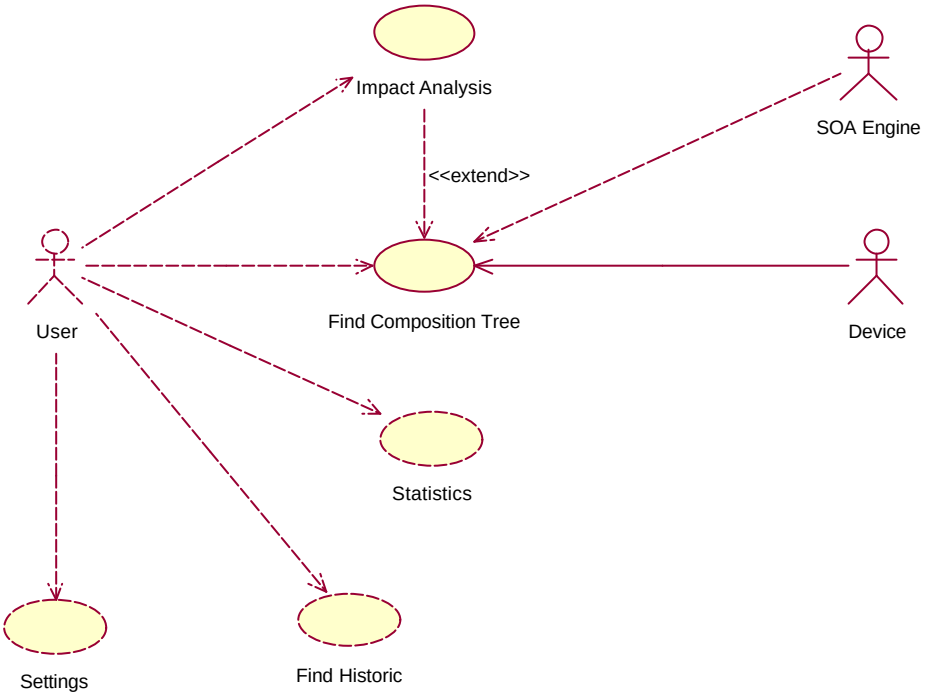

Figura 4.8 – Diagrama de Caso de Uso

Como descrito, foram definidos cinco requisitos a serem implementados, estes casos de uso são descritos abaixo:

- *Find Composition Tree*: caso de uso responsável pela funcionalidade de pesquisar e apresentar a árvore de composição que compõe um serviço. Deve realizar a partir de um serviço inicial uma busca em todos os servidores para listar os serviços da composição pesquisada;
- *Impact Analysis*: o caso de uso para calcular o impacto, executa a funcionalidade para buscar a árvore de composição, em posse desta estrutura é realizado o cálculo do impacto / complexidade da composição;
- *Find Historic*: esta funcionalidade deve armazenar e permitir consultas sobre as pesquisas já realizadas quanto a análise de impacto. Para cada execução da análise de impacto são armazenadas informações, como por exemplo, impacto calculado, e total de nodos da composição;
- *Statistics*: a partir das informações geradas no histórico, após um determinado número de execuções são geradas estatísticas das execuções do cálculo de impacto, tais como média do valor calculado, média de confiabilidade;
- *Settings*: parâmetros utilizados na fórmula ou parâmetros de execução do programa podem ser configurados de acordo com o perfil do usuário que o está utilizando, a partir dos requisitos deste caso de uso.

4.4.2 Modelagem Lógica e Física

Para o desenvolvimento da ferramenta optou-se por uma divisão em camadas para uma separação entre o protótipo (interfaces) e funcionalidades da aplicação. Esta visão é apresentada na Figura 4.9, que exemplifica as quatro camadas criadas para a aplicação.

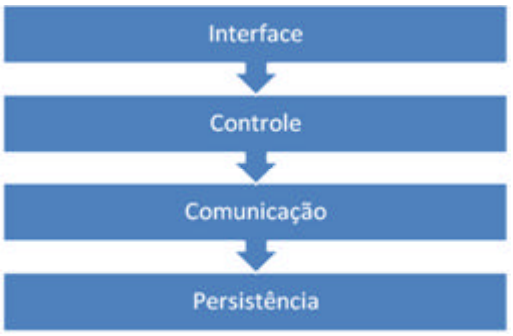

Figura 4.9 – Camadas que Compõe a Aplicação

A primeira camada “Interface” é responsável pela apresentação e interação com usuário, podendo esta ser facilmente substituída ou adaptada de acordo com as necessidades mantendo-se as funcionalidades do *Kernel* que se encontra nas três camadas subsequentes. Assim o *kernel* do sistema pode ser estendido para utilização em diferentes ferramentas, como a utilização em ferramentas de *profiling*, por exemplo.

Profiling são normalmente ferramentas incorporadas às IDE’s de desenvolvimento e possuem como principal funcionalidade verificar em quais locais (funções, rotinas, métodos ou serviços) um determinado sistema gasta a maior parte de seu tempo de processamento. Estas ferramentas podem ser usadas para medir as características de desempenho de um aplicativo em desenvolvimento [WILSON e KESSELMANN, 2000]. Assim, o desenvolvedor pode procurar otimizar os pontos com maior custo. No contexto do trabalho, este visa identificar quais os melhores serviços para que possam fazer parte da composição, a fim de melhorar o desempenho da mesma.

Para cada camada da aplicação foram implementados um conjunto de classes, e estas por sua vez separadas em pacotes por uma divisão lógica, esta divisão é apresentada no o diagrama de pacotes da Figura 4.10

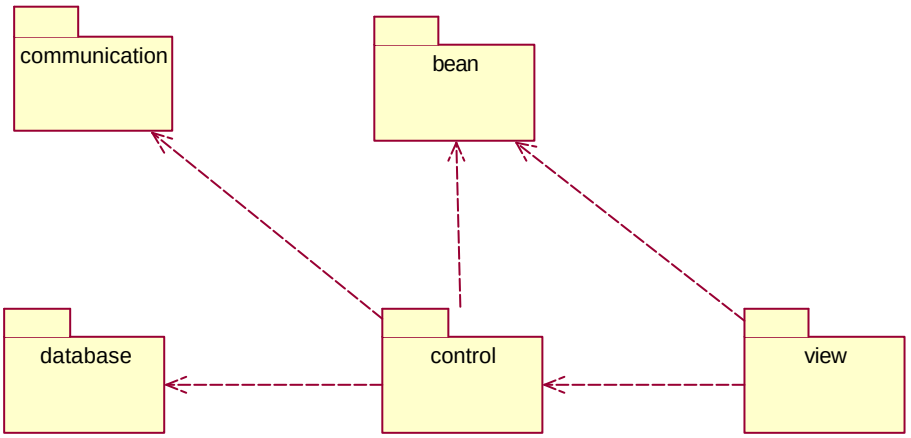

Figura 4.10 – Diagrama de Pacotes

Para uma melhor divisão e entendimento da aplicação, optou-se por separar as classes de acordo com sua funcionalidade em pacotes (*packages*). Estes pacotes são descritos a seguir:

- *view*: neste pacote são armazenadas as classes responsáveis por fazer a interface com o usuário, onde as telas são modeladas. As classes desta camada não devem conter regras de negócio, apenas repassar as interações do usuário para a camada de controle e apresentar-lhe as informações;
- *control*: nas classes deste pacote devem ser aplicadas as regras de negócio referentes a cada transação. Depois de aplicadas as pré-condições aos métodos as classes desta camada são responsáveis por chamar as classes que farão acesso ao banco de dados (BD). Quando necessário realizar a comunicação através de *web services*, sendo esta comunicação sempre realizada através da camada de controle;
- *communication*: pacote onde estão contidas as classes que realizam a comunicação através de *web services* com outros dispositivos;
- *bean*: neste pacote serão armazenadas as classes referentes às entidades do sistema (estas classes armazenam o conhecimento sobre estas entidades);
- *database*: o pacote database armazena as classes responsáveis por fazer a persistência em banco de dados. Todo acesso ao BD deve ser feito a partir desta camada, porque a partir desta camada, as classes estendem uma classe de uso genérico para acesso ao BD, a DataBase.java, que auxilia na comunicação com o BD da aplicação.

A divisão dos pacotes e desenvolvimento das classes foi baseado no padrão de projetos MVC (*Model-View-Controller*) [BUSCHMANN et. al., 1996], [PREE, 1995], [METSKER, 2004]. Assim separando a camada de apresentação das camadas de controle e persistência, como apresentados na divisão dos pacotes. Ao se seguir um padrão de projetos já consolidado no mercado, se têm como referência projetos já implementados o que permite seguir casos de sucesso e prever erros já identificados e documentados em projetos que seguem o mesmo modelo.

Após a definição dos pacotes, iniciou-se o desenvolvimento das classes as dividindo de acordo com suas funcionalidades. O diagrama de classes apresentado na Figura 4.11 representa uma abstração dos requisitos apresentados nos casos de uso da Figura 4.8 O diagrama apresenta tanto a camada de interface quanto as camadas de controle e persistência, e seus relacionamentos. Enquanto que o Anexo A apresenta o mesmo diagrama com os atributos e métodos para cada classe.

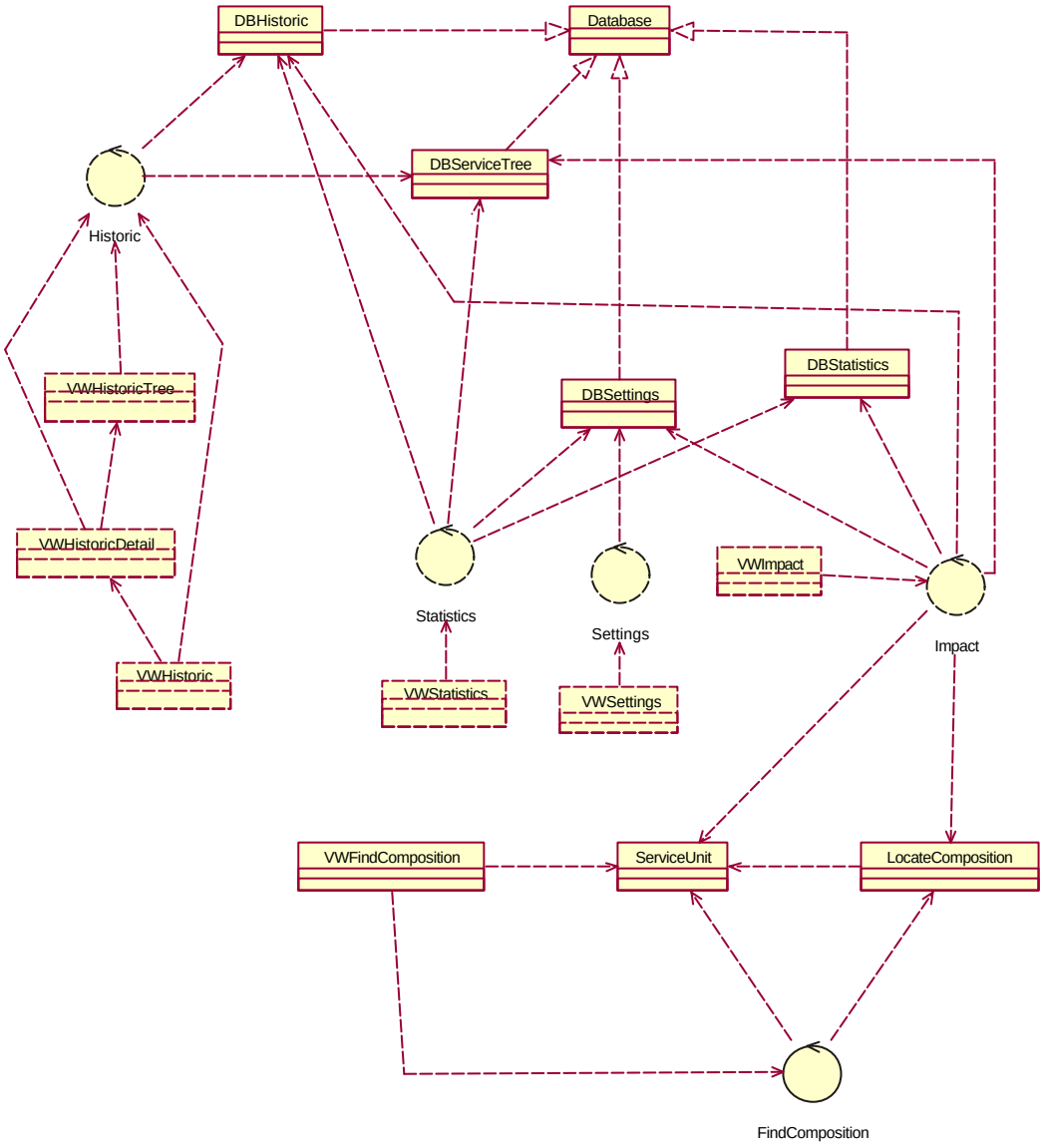

Figura 4.11 – Diagrama de Classes

As classes que constam no pacote *view* apresentadas no diagrama, representam as telas do sistema, que por sua vez fazem referência a arquivos XML que definem o *layout* de cada tela do sistema, desta forma para cada classe do pacote *view* existe um arquivo XML associado. Para cada transação do sistema, que pode ser composta por mais de uma tela, foi criada uma classe no pacote *control* que realiza a comunicação com a interface e aplica as regras de negócios existentes. As classes desta camada também são responsáveis por realizar a comunicação com a camada subsequente para manipulação dos dados.

Já as classes da camada de armazenamento realizam a persistência dos dados do sistema. Para cada tabela criada existe uma classe associada com os métodos necessários para sua manipulação. Todas as classes desta camada herdam da classe DataBase.java, que contém todos os métodos para criação e manipulação das tabelas.

Após o desenvolvimento da aplicação através de classes e arquivos XMLs, são gerados os componentes para sua distribuição. Os componentes gerados são apresentados na Figura 4.12, onde é demonstrada a divisão para distribuição e instalação.

Para permitir a reutilização de módulos da ferramenta, a aplicação foi dividida em três componentes, o que permite que as transações funcionem em separado da interface, permitindo sua reutilização em diferentes tipos de aplicação.

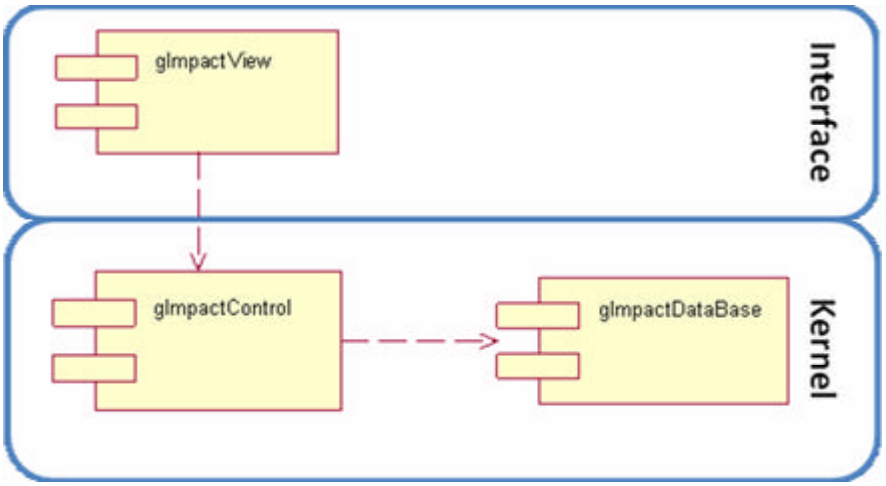

Figura 4.12 – Diagrama de Componentes

O aplicativo foi dividido em três componentes, sendo que o componente "gImpactView" representa a camada de interface da aplicação, o que permite que possa ser removido ou alterado por outro componente. Permitindo assim a reutilização do *kernel* composto pelos componentes "gImpactControl" e "gImpactDataBase" para outros tipos de aplicativos, como o ferramentas de *profiling,* por exemplo. O componente "gImpactControl" abrange as camadas de controle e comunicação apresentadas na Figura 4.9, enquanto que o componente "gImpactDataBase" incorpora a camada de persistência da aplicação.

4.4.3 Modelagem do Banco de Dados

Para controle da aplicação, como configurações e armazenamento do histórico por usuário, é utilizada uma base de dados local para cada dispositivo. Esta base de dados local visa personalizar o aplicativo em determinados pontos que permitam sua configuração e manter um histórico e estatísticas dos serviços que o usuário utiliza.

A base de dados foi criada sob a tecnologia SQLite [SQLite, 2008], o que permite sua utilização para os dispositivos celulares que tenham como plataforma o Android, sendo já nativo para estes aparelhos. A modelagem da base de dados pode ser vista na Figura 4.13. O banco de dados é criado em tempo de execução, na primeira vez que o usuário executa o programa.

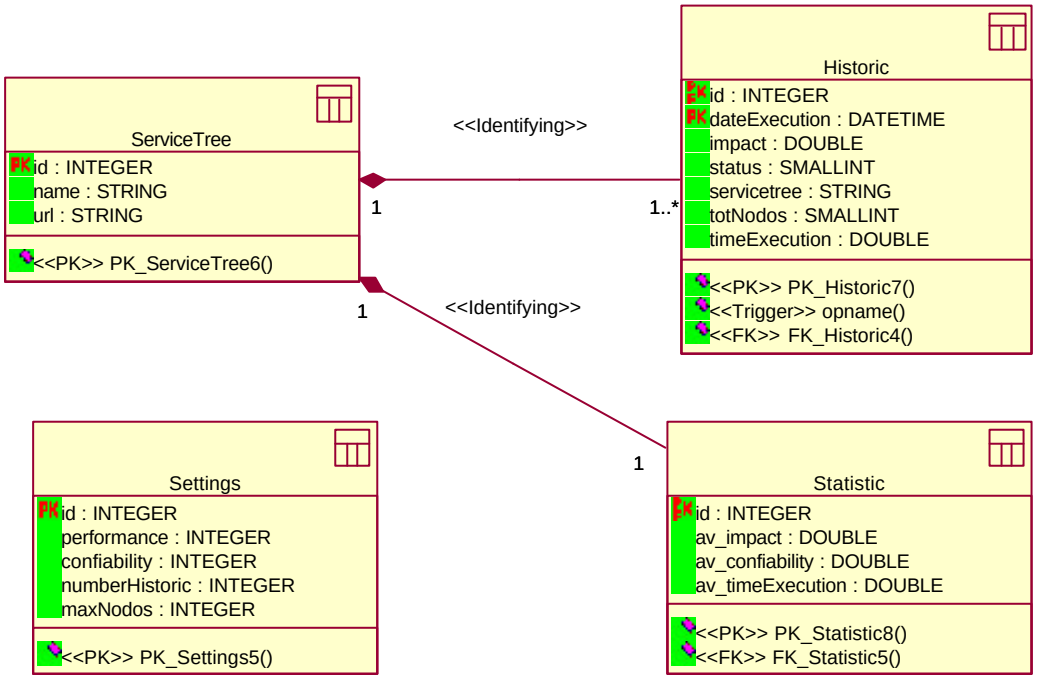

Figura 4.13 – Diagrama Entidade Relacionamento

A modelagem foi dividida em quatro tabelas, citadas abaixo:

- *ServiceTree*: tabela responsável por armazenar todas as composições já analisadas pelo usuário, armazenando apenas as informações básicas de cada composição;

- *Historic*: para cada análise em uma composição é gerado um registro na tabela "*Historic*". Nela estão contidas as informações referentes ao momento da análise de uma composição, pois esta poderá sofrer alterações ao longo de sua vida. A tabela *"Historic"* possui um relacionamento com a tabela *"ServiceTree"*, para cada registro nesta tabela deverá existir ao menos um registro na tabela *"Historic"*ou mais de um;
- *Statistic*: as estatísticas geradas das composições são armazenadas após um número pré-definido de execuções, depois de alcançado este número são geradas as estatísticas realizando uma média do cálculo de impacto e da disponibilidade armazenadas no histórico para cada composição. As estatísticas são sempre atualizadas, assim existirá apenas uma para cada composição da tabela "*ServiceTree*", desde que se tenha atingido o número mínimo de análises desta composição;
- *Settings*: esta tabela armazena as configurações por usuário. Permite configurar por exemplo o número de análises necessárias de uma composição para que seja gerada suas estatísticas. Além deste parâmetro também é permitida a configuração de algumas variáveis que serão utilizadas na fórmula para cálculo de impacto.

4.4.4 Pesquisa da Árvore de Composição

A primeira questão a ser respondida para análise de impacto, é de quais serviços fazem parte de uma composição. Um arquivo WSDL expõe apenas a interface de entrada e saída de um serviço, não permitindo assim que seja recuperada a árvore de composições. Para permitir a identificação desta composição é utilizada a linguagem inSOA [ZANUZ et al., 2008], que apresenta quais serviços fazem parte do primeiro nível da composição, que a partir destes é possível a descoberta dos demais

A partir da exposição do primeiro nível de composição é possível a busca de toda a árvore, buscando para cada serviço sua composição e ao final se obter a composição completa, a Figura 4.14 exemplifica esta busca.

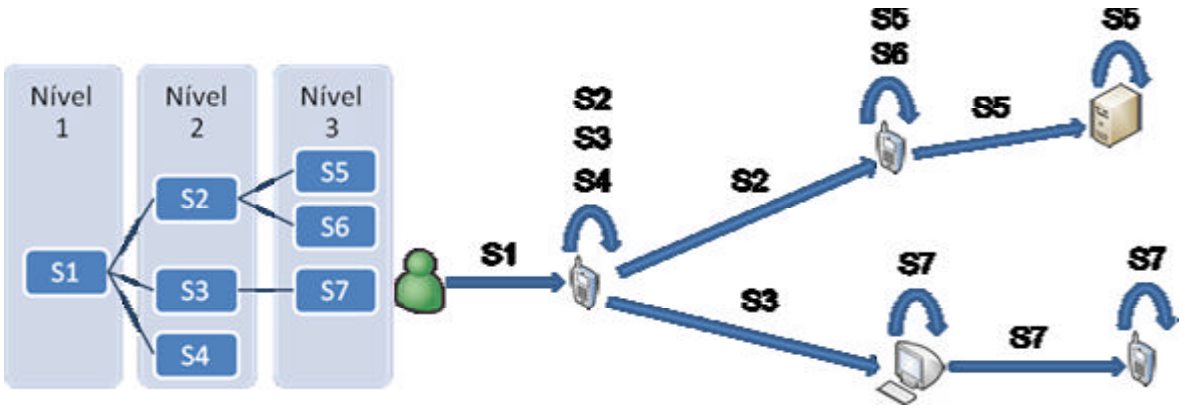

Figura 4.14 – Busca da Árvore de Composições

Para montar a árvore de composições apresentada na figura, é necessária uma busca em todos os serviços que a compõe, independente do tipo de dispositivo, podendo ser um desktop ou um dispositivo móvel com a plataforma Android, porém deve ser ressaltado que as composições devem estar escritas na linguagem inSOA. As composições expostas através de um WSDL serão consideradas como último nível de *web services*, pelo fato de um WSDL não expor a composição. Assim, a função busca em uma primeira execução os serviços que encontram-se no nível 1, a partir deste os serviços são analisados novamente e se busca a composição de cada serviço, caso este não seja um *web services*, o que caracteriza o ponto de parada do serviço e último nível de uma composição. Desta forma a função é executada de forma recursiva até que sejam analisados todos os serviços de uma composição. Para exemplificar as interações e trocas de mensagens entre os métodos é utilizado o diagrama de sequência [FURLAN, 1998], apresentado na Figura 4.15.

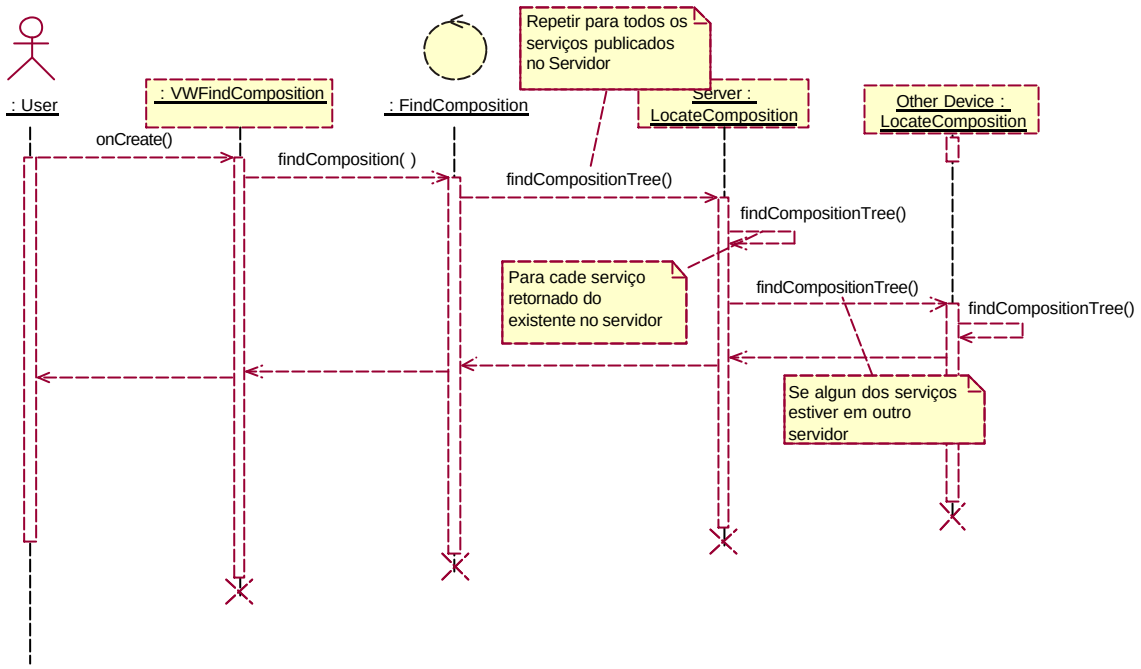

Figura 4.15 – Diagrama de Seqüência de Busca da Composição

A sequência para busca da árvore de composições, no protótipo é iniciada através da interface que por sua vez executa um método da camada de controle. A partir desta camada a primeira ação é pesquisar os serviços que se encontram no próprio dispositivo, este procedimento é executado até que não existam mais serviços no próprio servidor, caso a árvore não esteja completa então o método é executado novamente porém enviando a requisição para um outro servidor onde se encontram os demais serviços. Desta forma para cada composição encontrada, os serviços que fazem parte são pesquisados independentes do servidor em que se encontram. O método é executado de forma recursiva até que encontre o seu ponto de parada, que no caso de uma composição corresponde a um serviço simples, ou seja um serviço que retorne um WSDL. Enquanto os serviços retornarem sua especificação na linguagem inSOA, indica que este é uma composição. O método que realiza a busca da árvore de composições é apresentado na Figura 4.16, onde é implementada uma chamada recursiva para que seja possível buscar todos os serviços que fazem parte da composição.

```
      public ServiceUnit findCompositionTree(ServiceUnit pService) throws
Exception {
            ServiceUnit ret = new ServiceUnit();
            //Busca XML do inSOA representando os serviços
            String serviceTree = new String();
            try {
                  serviceTree = this.returnComposition(pService.getName());
            } catch (Exception e) {
                  throw new Exception("Servico nao ativo");
            }
            //Busca do XML os serviços em um Vector
            Vector<ServiceUnit> vet = getServices(serviceTree);
            //Adiciona vector retornado ao Serviço pai
            if (vet.size() > 0) {
                  pService.setVectorComposition(vet);
            }
            //Percorre o Vector carregando a composição de cada serviço
            for (int i = 0; i < vet.size(); i++) {
                  ServiceUnit u = (ServiceUnit)vet.elementAt(i);
                  ret = findCompositionTree(u);
            }
            return pService;
      }
```

Figura 4.16 – Código para Pesquisa da Árvore de Composições

Após a geração da árvore esta é retornada pela classe do pacote *control*, manipulada e apresentada de acordo com a ferramenta. O protótipo que implementa esta funcionalidade é apresentado na Figura 4.17.

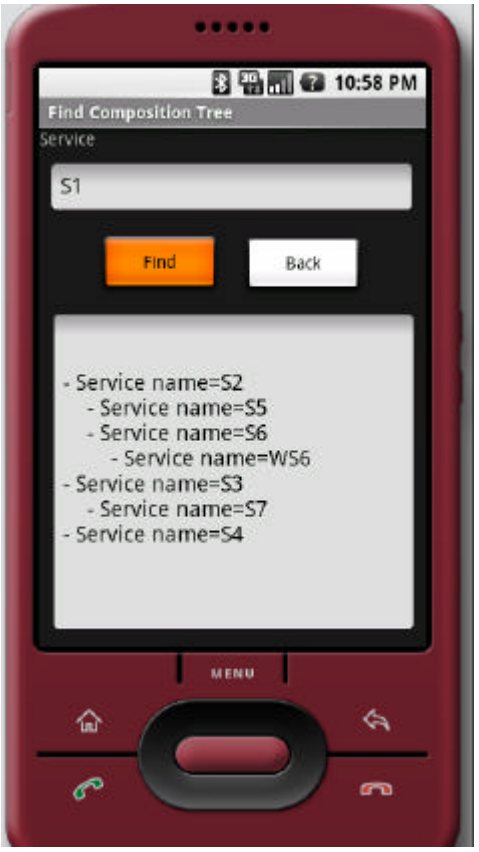

Figura 4.17 – Tela Pesquisa Árvore de Composições

No protótipo o usuário deve informar qual composição (serviço) se quer pesquisar e após a busca de toda a árvore executando a sequência da Figura 4.15 ela é apresentada em uma forma textual para visualização. Esta funcionalidade no protótipo visa apenas a consulta da árvore de composição, porém a funcionalidade é também utilizada para o cálculo de impacto.

4.4.5 Cálculo de Impacto

Com a questão da montagem da árvore de composições implementada, é necessário inferir a fórmula proposta para o cálculo de impacto. Para isto foi desenvolvida uma transação que aplica a fórmula sobre a árvore de composições. A tela que mostra os resultados desse cálculo é apresentada na Figura 4.18.

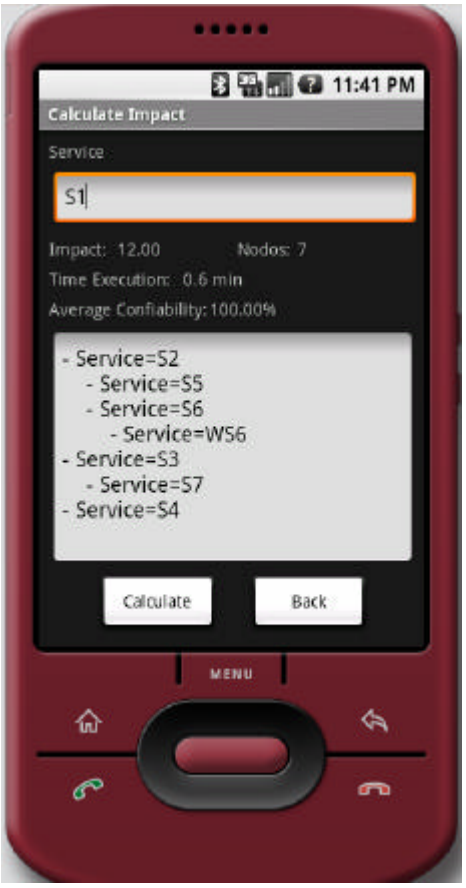

Figura 4.18 – Tela para Cálculo de Impacto

Nesta transação é necessário que o usuário informe qual o serviço em que se pretende realizar a análise de impacto. Para pesquisa na ferramenta o serviço já deve estar implantado, porém caso o componente para análise seja utilizado em uma ferramenta de *profiling*, basta que a composição de primeiro nível esteja na ferramenta para que se busque o restante da árvore de composições.

Ao se digitar o serviço então é executada a busca da árvore de composições que é apresentada em tela para visualização do usuário, com isto a fórmula proposta é executada. Além do resultado do cálculo, são apresentados em tela o total de nodos ou serviços que fazem parte da composição, o tempo de execução e a média de confiabilidade do serviço, caso esta já tenha sido calculada nas estatísticas. Para cada execução do cálculo é gravado um registro na tabela de histórico para permitir posteriormente comparações quanto a evolução do serviço e que sejam geradas estatísticas para melhora de performance nas composições.

4.4.6 Histórico

Para cada execução do cálculo de impacto, as informações dos serviços são armazenadas em uma tabela de histórico. São armazenadas informações como data e hora de execução, o valor calculado de impacto do serviço naquele momento, se o serviço estava ativo, o número de serviços que faziam parte da composição e a própria árvore de composições. Assim é possível que o usuário consulte posteriormente dados das execuções e permite que sejam geradas estatísticas referentes aos serviços. As telas da transação de histórico podem ser vistas na Figura 4.19.

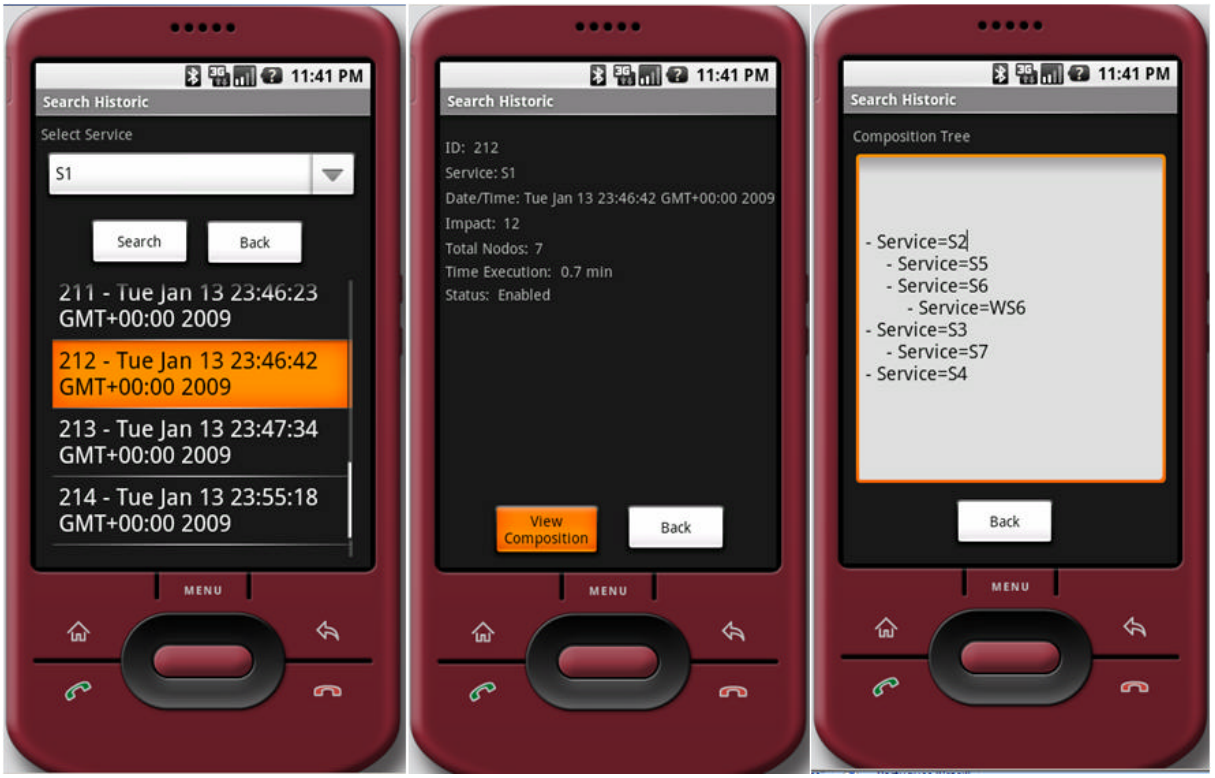

Figura 4.19 – Telas para Consulta de Histórico

A transação para consulta de históricos foi dividida em três telas: a primeira tela apresenta os serviços que já foram executados na ferramenta para seleção, depois de selecionado o serviço são apresentadas em uma lista todas as execuções do serviço, ordenadas pelo código do histórico e data/hora de execução. Após a seleção de uma execução específica então é apresentada a segunda tela, onde são apresentadas informações sobre a execução do serviço, como por exemplo, o status do serviço, o total de nodos que faziam parte da composição, o valor calculado de impacto. Se o usuário optar pela visualização da árvore de

composições, então é apresentada a terceira tela onde pode se visualizar toda a composição do serviço.

4.4.7 Configurações

Para permitir que a aplicação seja mais aderente as necessidades do usuário, é possível que alguns dos valores utilizados no sistema sejam parametrizáveis. Estes valores são utilizados na fórmula para cálculo do impacto e para a geração de estatísticas dos serviços, a tela para configuração é apresentada na Figura 4.20.

Figura 4.20 – Tela para Ajuste de Configurações

A tela de configurações possui dois campos indicando o percentual de importância que o usuário define para a confiabilidade do serviço e desempenho, a soma destes deve ser igual a 100%. Estes percentuais serão utilizados na fórmula para cálculo de impacto, tornando assim a fórmula mais adaptada as necessidades de cada usuário do sistema. Outro valor parametrizável que é utilizado durante a pesquisa da árvore de composição é a profundidade

máxima a ser pesquisada, evitando assim que um serviço entre em *loop*. Por fim é possível alterar o número de execuções necessárias de um serviço para que sejam geradas estatísticas do mesmo, assim para um usuário que utiliza pouco a execução de serviços possa configurar para assim mesmo sejam geradas estatísticas quanto a sua execução.

4.4.8 Estatísticas

As estatísticas são geradas a partir de um determinado número de execuções de um serviço, o número de execuções é configurado pelo usuário. Quando a transação é acionada é realizada uma verificação na tabela de configurações (*Settings*) de quais serviços já atingiram o número especificado para geração de estatísticas, com isto para as transações que atingiram é realizada uma sumarização de todos os dados da tabela de histórico (*Historic*) gerando estatísticas quanto a média de impacto do serviço e média de confiabilidade. Esta interação pode ser vista no diagrama de sequência da Figura 4.21.

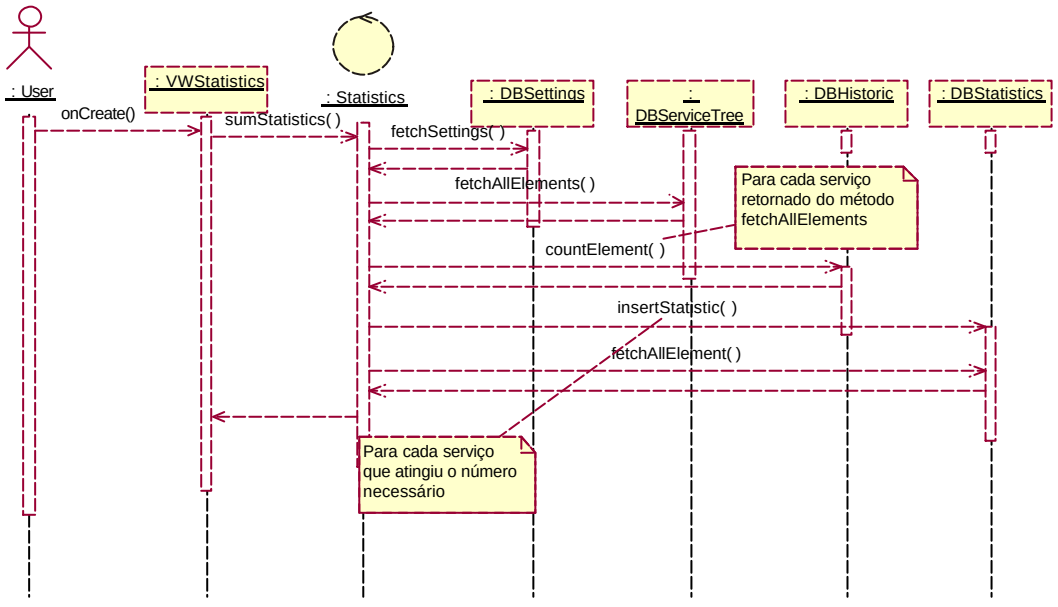

Figura 4.21 – Diagrama de Sequência para Geração de Estatísticas

Na sequência apresentada é possível visualizar toda interação desde a chamada da tela onde se inicia a transação, até as sumarizações e apresentação das estatísticas. A Figura 4.22 apresenta a tela de consulta.

Figura 4.22 – Tela para Consulta de Estatísticas

Nesta tela são carregados todos os serviços que foram geradas estatísticas em uma lista para seleção. Ao se selecionar o serviço específico então são preenchidos os campos com a média do impacto, média de confiabilidade do serviço, que é utilizada na fórmula para o cálculo e a média do tempo de execução.

O próximo capítulo apresenta três estudos de caso para demonstrar a utilização da ferramenta gImpact implementada neste trabalho e sua reutilização em outros aplicativos.

5 ESTUDOS DE CASO

Este capítulo tem como objetivo apresentar três cenários possíveis para utilização da ferramenta gImpact e uma avaliação de sua utilização nos cenários demonstrados. Procura-se abordar questões quantitativas em sua utilização.

Os estudos de caso que auxiliam na validação deste trabalho foram realizados em conjunto com outros dois trabalhos que foram implementados por nosso grupo de pesquisa, conforme ilustra a figura 5.1. O fluxo de funcionamento do estudo de caso é o seguinte:

1. Usuário edita uma composição escrita através da linguagem inSOA;
2. Usuário compila o script através de uma ferramenta do pacote inSOA;
3. Usuário ajusta eventuais erros;
4. Através de uma ferramenta do pacote inSOA, usuário gera um código XML referente à composição de serviços;
5. Usuário publica a composição em SmallSOA;
6. Usuário busca a composição em SmallSOA;
7. Usuário executa a composição em SmallSOA;
8. A ferramenta gImpact localiza a composição em SmallSOA e efetua suas análises;
9. A ferramenta gImpact executa a composição para obtenção dos tempos de execução para estatísticas;
10. A partir das análises feitas pela ferramenta gImpact, usuário altera o script inSOA e o fluxo recomeça.

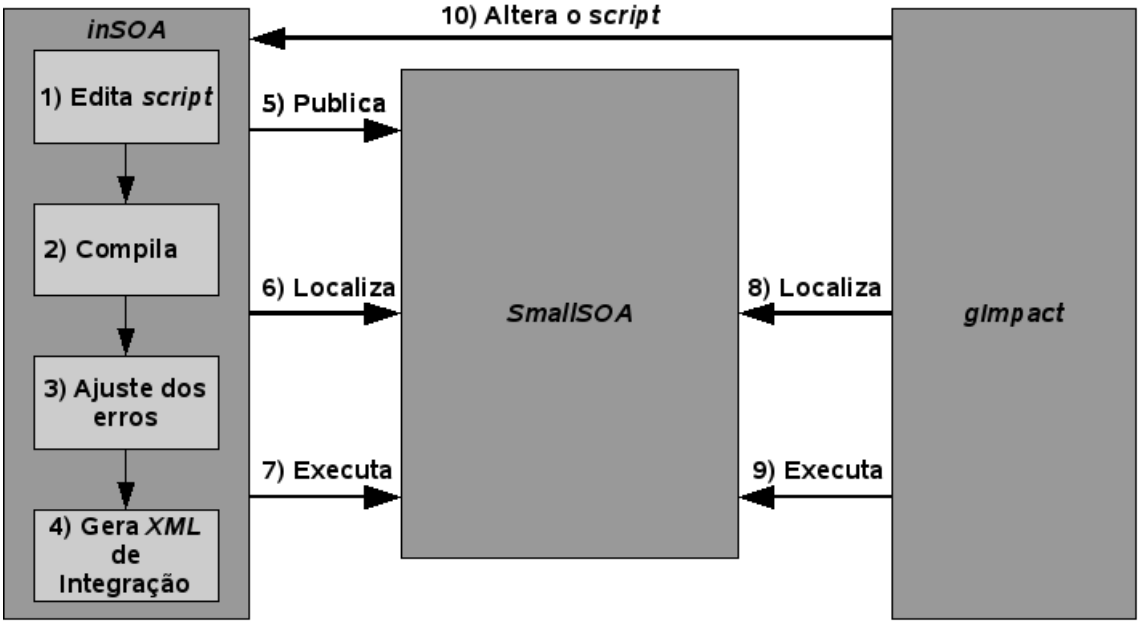

Figura 5.1 – Fluxo dos estudos de caso

O foco dos estudos de caso apresentados correspondem aos passos 8, 9 e 10. Onde são descritas as funcionalidades da ferramenta gImpact.

5.1 Configuração do Ambiente de Execução

Para os dois primeiros cenários apresentados (Consulta de Livros e Pesquisa para Viagem) foi elaborado um ambiente com emuladores para o celular Android, em uma rede interna, para a execução das composições descritas na linguagem inSOA, com acesso a Internet para que sejam executados os *web services*, serviços estes que serão externos a rede. O ambiente de testes é apresentado na Figura 5.2.

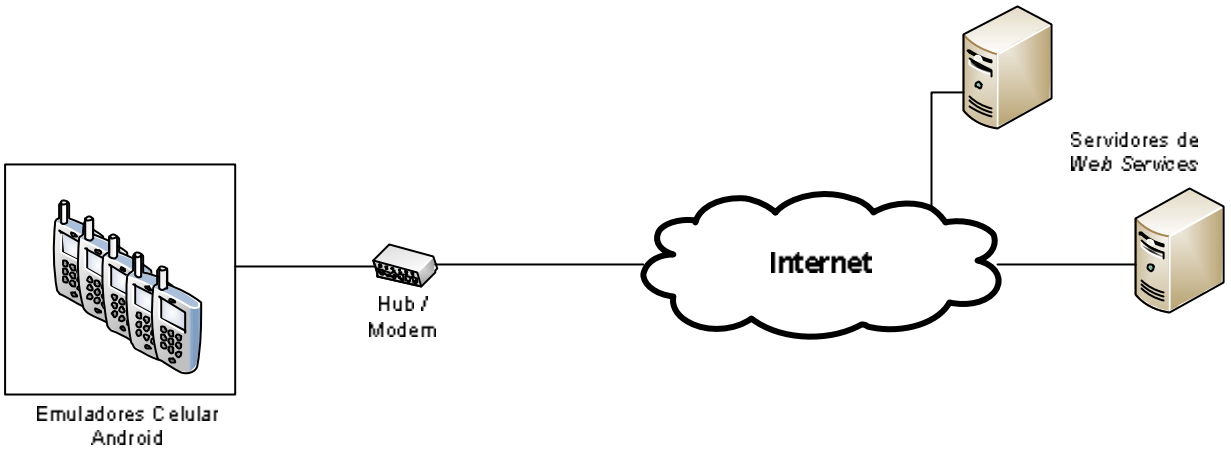

Figura 5.2 – Ambiente de Execução

Neste ambiente, cada composição encontra-se em um celular distinto, assim o número de emuladores ativos pode variar de acordo com o estudo de caso em utilização. Enquanto que os *web services* estão localizados em servidores externos, na Internet. Ao se utilizar de *web services* disponíveis na Internet procura-se representar a utilização de composições em servidores distribuídos, estando sujeito as oscilações decorrentes da uma rede deste tipo, fornecendo assim informações mais consistentes quanto a disponibilidade e mudanças que possam vir a acontecer nos serviços.

Nesta configuração também é possível validar a utilização da infra-estrutura com um ambiente hibrido, com celulares e servidores *desktop*, sendo estes independente do sistema operacional em utilização, devido a interoperabilidade alcançada com a utilização de *web services*.

5.2 Cenário 1: Consulta Livros e Validação para Compra

Uma grande parte dos sites de venda de livros passou a disponibilizar serviços para que seus clientes possam consultar suas bases de dados através de aplicações próprias ou através de composições de seus serviços. Assim este cenário se utiliza de *web services* que contenham uma relação com a venda de livros. A composição visa realizar uma pesquisa de livros a partir de um título na base de dados da Amazon [AMAZON WS, 2008], e realizar a conversão para a moeda corrente de seu valor, a partir desta seleção já deve ser realizada uma validação dos dados do comprador, como por exemplo, número de cartão de crédito,

validação no e-mail do usuário e validações do CPF, para posterior compra. A estrutura da composição pode ser vista na Figura 5.3.

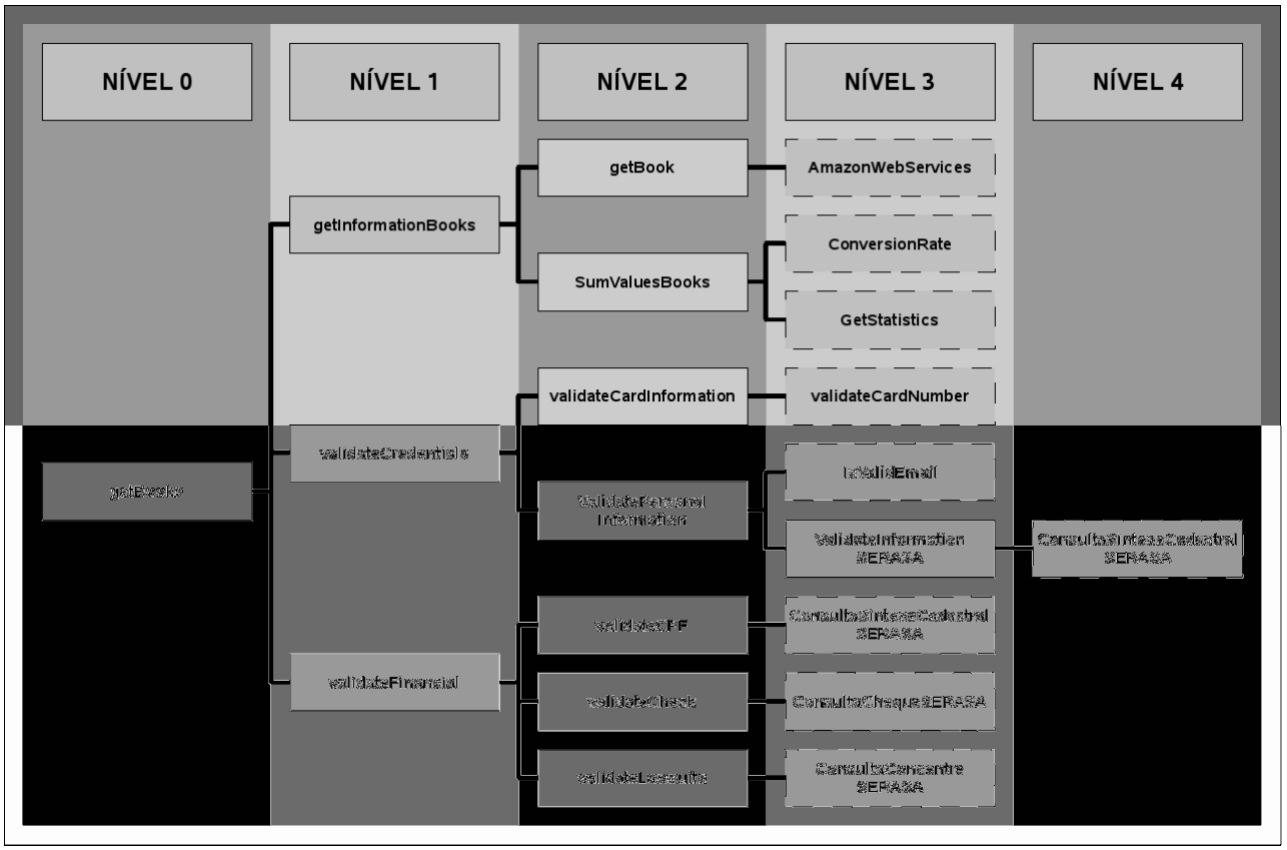

Figura 5.3 – Árvore de Composição (Consulta de Livros)

Neste cenário, as composições foram descritas na linguagem inSOA, enquanto que os *web services* são de fornecedores externos implementados na linguagem .NET [MSDN FRAMEWORK, 2008] e Java [JAVA, 2008]. Foram elaboradas 12 (doze) composições e utilizados 9 (nove) *web services* para testes no cenário apresentado. O Quadro 5.1 apresentado descreve os *web services* externos utilizados na composição.

Quadro 5.1 – Web Services Utilizados na Consulta de Livros

Descrição	***Web Service***
Buscar lista de livros	http://soap.amazon.com/AmazonSearchService
Realiza conversão da moeda	http://www.webservicex.net/CurrencyConvertor.asmx/ ConversionRate
Soma os valores dos livros selecionados	http://www.webservicex.net/Statistics.asmx/ GetStatistics
Validar Cartão de crédito	http://www.webservicex.net/CreditCard.asmx/ ValidateCardNumber
Validar e-mail	http://www.webservicex.net/ValidateEmail.asmx/ IsValidEmail
Consulta Cheques por CPF/CNPJ	http://www.consultacpf.com/webservices/consultacpf.asmx/ ConsultaChequeSERASA
Consulta CPF/CNPJ CONCENTRE SERASA – Ações, BACEN	http://www.consultacpf.com/webservices/consultacpf.asmx/ ConsultaConcentreSERASA
Sintese Cadastral SERASA	http://www.consultacpf.com/webservices/consultacpf.asmx/ ConsultaSinteseCadastralSERASA

Para os *web services*, que representam as folhas da árvore, pois são a menor unidade de uma composição, foram utilizados serviços disponibilizados na Internet. Para demonstrar a integração e viabilidade de utilização da ferramenta com o ambiente que se encontra em utilização no mercado.

5.2.1 Configuração da Ferramenta gImpact

Para realização dos testes, a ferramenta gImpact foi configurada conforme a Tabela 5.1, onde as variáveis podem ser configuradas de acordo com as preferências do usuário através de interface apropriada.

Tabela 5.1 – Configurações gImpact (Consulta de Livros)

Variável	**Valor**
Núm. para geração de histórico	10
% Confiabilidade	90
Profundidade da árvore	8

Após a décima execução da análise de impacto, conforme parâmetros configurados serão geradas estatísticas dos serviços o que será considerado na fórmula para o cálculo, visando calibrar a fórmula de acordo com as preferências do usuário. Para este cenário foi

considerada como importante a confiabilidade dos serviços, assim a variável foi configurada para 90% (noventa) para obter maior impacto na fórmula. A profundidade foi configurada para leitura no máximo em 8 (oito) níveis, evitando que a procura de toda a árvore leve um tempo excessivo.

5.2.2 Resultado dos Testes

Após a configuração do ambiente e dos parâmetros do sistema foram realizadas execuções para consultar o impacto e demais informações obtidas através da ferramenta gImpact da composição elaborada para o estudo de caso. Estes dados são apresentados na Tabela 5.2.

Para coleta dos dados, o programa foi executado entre 9:00 (nove) horas e 18:00 (dezoito) horas, sendo que para cada horário foram executadas dez consultas. Desta forma, a tabela a seguir apresenta a média dessas execuções.

Tabela 5.2 – Resultado de Resposta para Análise de Impacto (Consulta de Livros)

	09hs	**10hs**	**11hs**	**12hs**	**13hs**	**14hs**	**15hs**	**16hs**	**17hs**	**18hs**
Impacto (média)	68	68	68,40	68,67	68,86	68,96	69,05	69,08	69,09	69,10
Nodos	20	20	20	20	20	20	20	20	20	20
Tempo Execução (seg)	28,121	27,015	27,813	27,586	28,045	28,459	28,419	28,687	28,952	29,057
Falhas	0	3	2	2	1	2	0	0	3	2
% Conf. (média)	100	85	83,33	82,50	84,00	83,33	85,92	87,65	85,71	85,15

Como apresenta a tabela, o total de nodos não sofreu alteração ao longo dos testes, pois não ocorreram alterações na composição do estudo de caso, porém esta informação se torna importante uma vez que ao longo do tempo os serviços possam sofrer alterações e por consequência isso implicaria em seu custo. Para o cálculo da média do impacto e confiabilidade do serviço, a ferramenta contabiliza todas as execuções do serviço, como configurado na ferramenta para que a cada 10 (dez) execuções sejam geradas estatísticas, assim todas as horas subsequentes a primeira, utilizarão toda a base de execuções para seus cálculos. A variação do valor do impacto é apresentada na Figura 5.4.

Figura 5.4 – Gráfico Impacto x Tempo de Execução

O fato de não ocorrerem alterações nas composições durante os testes, a principal variável que poderia influenciar no cálculo de impacto é a confiabilidade dos serviços. Assim, o gráfico apresenta um aumento do valor calculado a medida que ocorrem falhas na execução dos serviços, reduzindo assim sua confiabilidade. As médias de confiabilidade que influenciam no valor calculado são apresentadas no gráfico da Figura 5.5.

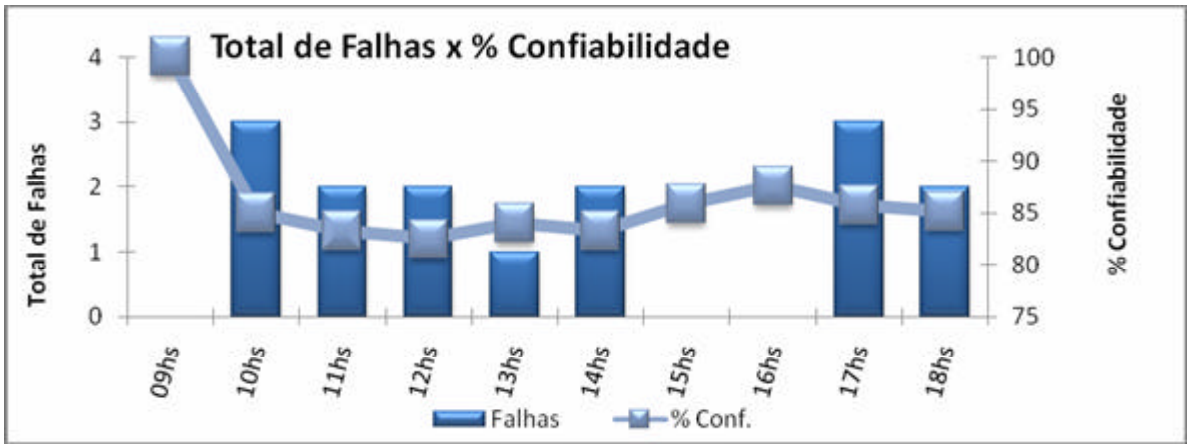

Figura 5.5 – Gráfico Total de Falhas x % Confiabilidade

Neste gráfico é apresentado para cada hora o total de falhas em dez execuções e a média geral de confiabilidade do serviço. Assim, como apresentado a medida em que ocorrem falhas na execução do serviço a média de confiabilidade é reduzida, influenciando no cálculo de impacto do serviço. Este impacto na fórmula pode ser vista na Figura 5.6 onde o gráfico apresenta uma relação entre estas variáveis.

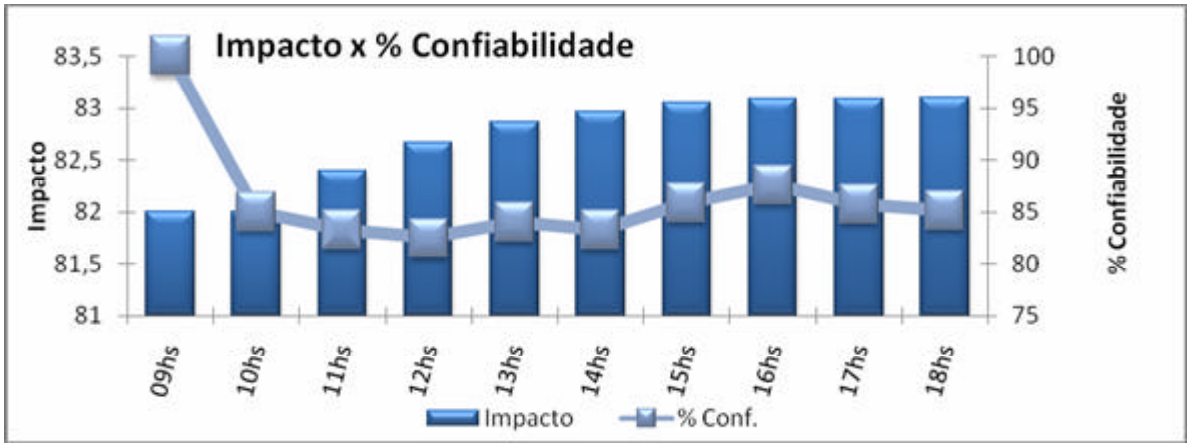

Figura 5.6 – Gráfico Impacto x % Confiabilidade

Neste gráfico é possível visualizar que a medida que a confiabilidade dos serviços diminui o impacto calculado aumenta. Deve-se observar que na fórmula, para a confiabilidade é utilizado o valor calculado nas estatísticas, assim este valor só é alterado a cada dez execuções, conforme configuração realizada na seção 5.2.1. Assim, na hora subsequente ao cálculo da média de confiabilidade que esta vai influenciar na fórmula.

5.3 Cenário 2: Pesquisa para Viagem

O segundo cenário apresentado corresponde a uma composição realizada para pesquisar dados referentes a uma viagem, visando disponibilizar ao usuário informações do país que se pretende visitar. Diversas consultas, como serviços de meteorologia, condições de aeroportos, tráfego rodoviário e conversões de moedas estão disponíveis na Internet. Assim, é possível que se crie composições de acordo com as informações que se julguem pertinentes a uma viagem. Os serviços que fazem parte desta composição e sua estrutura podem ser vistas na Figura 5.7.

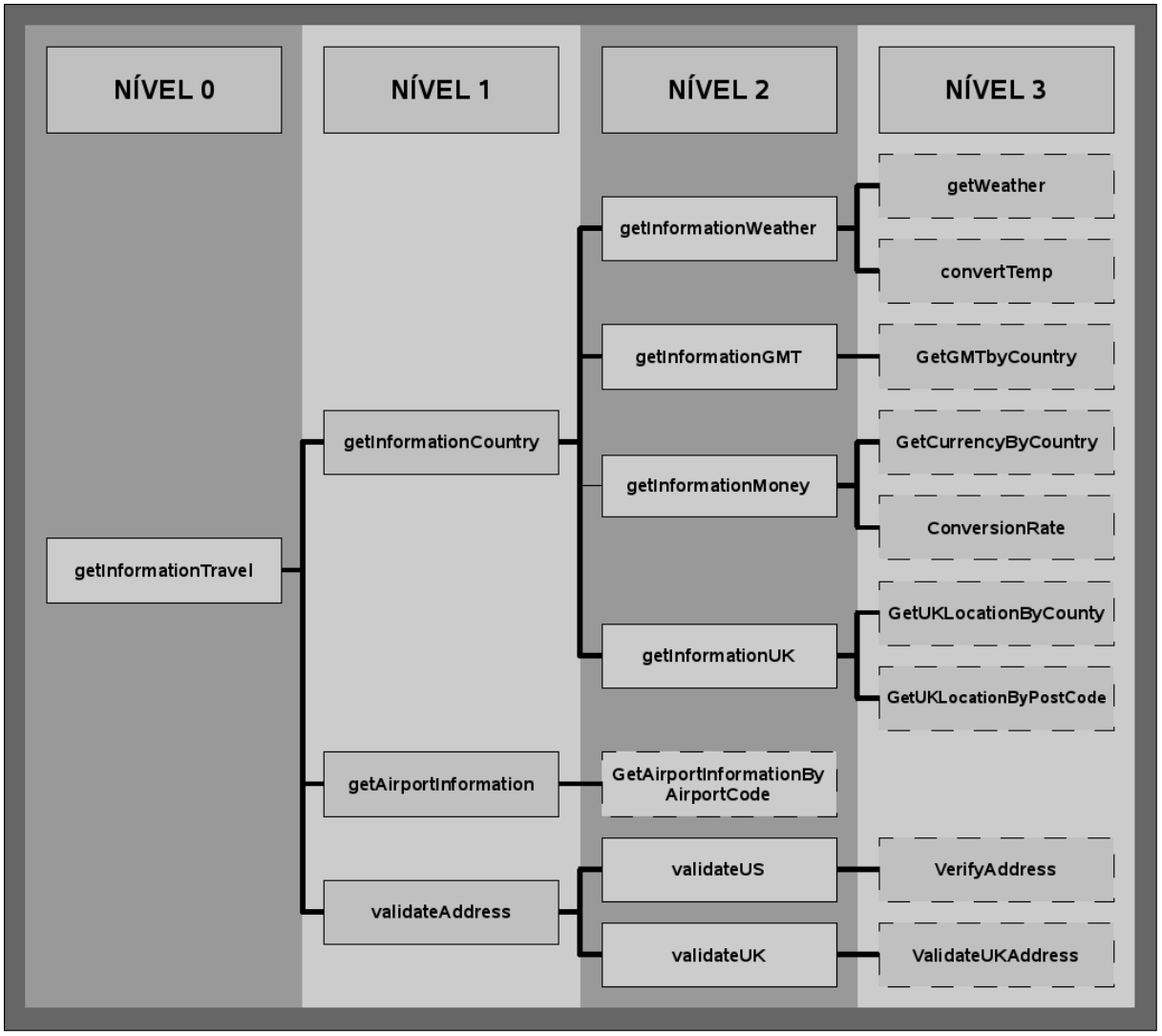

Figura 5.7 – Árvore de Composição (Pesquisa Viagem)

Para este cenário, as composições foram implementadas na linguagem inSOA, em um total de 10 (dez), e cada composição encontra-se em um emulador de celular, representando 10 (dez) celulares neste cenário. Os demais serviços correspondem a *web services*, sendo um total de 10 (dez), que se encontram em servidores na Internet implementados em .NET. Os *web services* utilizados são descritos no Quadro 5.2.

Quadro 5.2 – Web Services Utilizados na Pesquisa de Viagem

Descrição	***Web Service***
Busca a previsão do tempo da cidade	http://www.webservicex.net/globalweather.asmx/ GetWeather
Busca conversão de temperatura	http://www.webservicex.net/ConvertTemperature.asmx/ ConvertTemp
Busca informações sobre o horário do país	http://www.webservicex.net/country.asmx/ GetGMTbyCountry
Realiza conversão da moeda	http://www.webservicex.net/CurrencyConvertor.asmx/ ConversionRate
Busca a moeda do País	http://www.webservicex.net/country.asmx/ GetCurrencyByCountry
Informações sobre o aeroporto	http://www.webservicex.net/airport.asmx/ getAirportInformationByCityOrAirportName
Valida endereço se for US	http://www.webservicex.net/usaddressverification.asmx/ VerifyAddress
Valida endereço se for UK	http://www.webservicex.net/uklocation.asmx/ ValidateUKAddress
Busca informações se for UK	http://www.webservicex.net/uklocation.asmx/ GetUKLocationByCounty http://www.webservicex.net/uklocation.asmx/ GetUKLocationByPostCode

Neste cenário alguns dos *web services* só serão executados de acordo com os parâmetros de entrada, pois executam consultas específicas de um determinado País. Porém, para cálculo de impacto, a árvore toda deve ser considerada, uma vez que deve ser considerado o pior caso para estimar seu custo.

5.3.1 Configuração da Ferramenta gImpact

Neste novo cenário a ferramenta foi configurada conforme a Tabela 5.3, para uma melhor adaptação das necessidades simuladas do usuário.

Tabela 5.3 – Configurações gImpact (Pesquisa para Viajem)

Variável	**Valor**
Núm. para geração de histórico	10
% Confiabilidade	50
Profundidade da árvore	8

Para este cenário foi considerada como baixa a importância em relação a confiabilidade dos serviços, assim a variável foi configurada para 50% (cinquenta). A profundidade foi configurada para leitura no máximo em 8 (oito) níveis, evitando que a procura de toda a árvore leve um tempo excessivo. O número para a geração de histórico foi configurado em 10 (dez).

5.3.2 Resultado dos Testes

Este cenário utiliza o mesmo ambiente elaborado para o estudo de caso de consulta de livros. Ajustando-se apenas as configurações do sistema. Após estes valores configurados foram realizadas as execuções para consultar o impacto e demais informações que são retornadas da ferramenta gImpact. Estes dados são apresentados na Tabela 5.4.

Para coleta dos dados, o programa foi executado entre 9:00 (nove) horas e 18:00 (dezoito) horas, sendo que para cada horário foram executadas dez consultas. Assim, a tabela a seguir apresenta a média dessas execuções.

Tabela 5.4 – Resultado de Resposta para Análise de Impacto (Pesquisa para Viajem)

	09hs	**10hs**	**11hs**	**12hs**	**13hs**	**14hs**	**15hs**	**16hs**	**17hs**	**18hs**
Impacto (média)	63	63,2	63,5	63,72	63,79	63,92	64,02	64,12	64,19	64,29
Nodos	19	19	19	19	19	19	19	19	19	19
Tempo Exec. (seg)	27,797	27,872	27,983	27,045	27,126	27,987	28,012	27,994	28,045	28,098
Falhas	1	4	3	3	6	4	5	4	6	2
% Conf. (média)	90	75	73,33	72,5	66	65	62,86	62,50	60	62

Da mesma forma que a composição do cenário 1, o total de nodos não sofreu alteração ao longo dos testes, pois não ocorreram alterações na composição do estudo de caso. Para o cálculo das médias apresentadas na tabela foram contabilizadas todas as execuções do serviço em questão, totalizando 100 (cem) execuções ao final dos testes, sendo geradas estatísticas a cada dez execuções conforme parâmetros do sistema. A variação do valor do impacto é apresentada no gráfico da Figura 5.8.

Figura 5.8 – Gráfico Impacto x Tempo de Execução

Apesar do número de falhas elevado na execução do serviço, a variação do resultado final do impacto é pequena, uma vez que foi parametrizada com uma importância baixa a confiabilidade dos serviços. O gráfico com a média de confiabilidade e falhas ocorridas é apresentado na Figura 5.9.

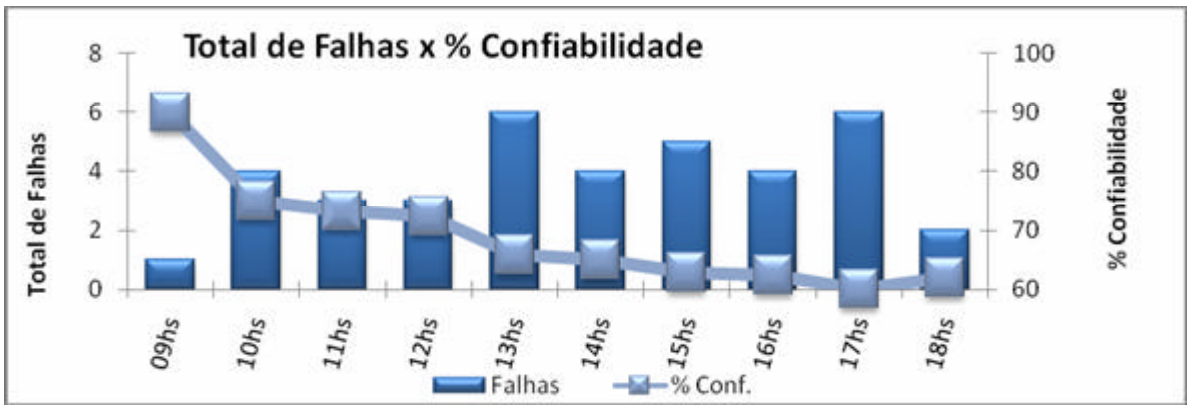

Figura 5.9 – Gráfico Total de Falhas x % Confiabilidade

O gráfico apresenta a média geral de confiabilidade do serviço e o total de falhas a cada hora, com um total de dez execuções por hora. Assim, como apresentado a medida que ocorrem falhas na execução do serviço a média de confiabilidade é reduzida, influenciando no cálculo de impacto do serviço. Este impacto na fórmula pode ser vista na Figura 5.10 onde o gráfico apresenta uma relação entre estas duas variáveis.

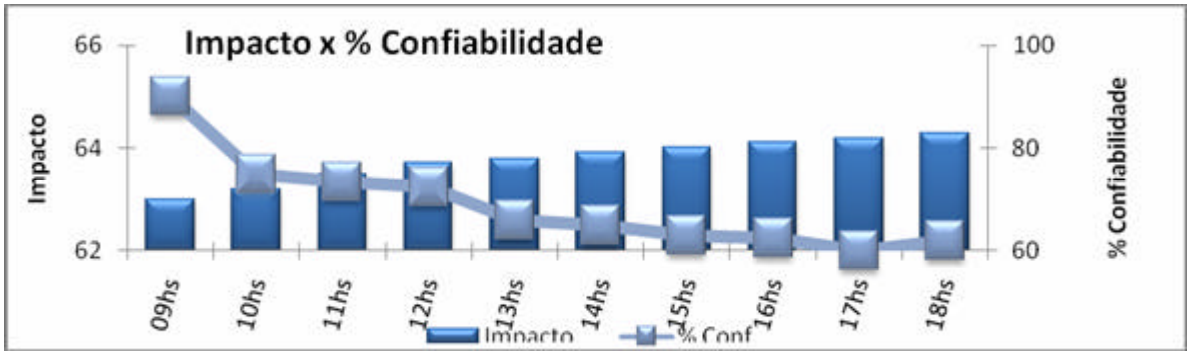

Figura 5.10 – Gráfico Impacto x % Confiabilidade

Neste gráfico é possível visualizar que à medida que a confiabilidade dos serviços diminui o impacto calculado aumenta, porém a variação do resultado é pequena devido ao fato de a configuração do peso da confiabilidade ter sido ajustada para um valor baixo. Deve-se observar que na fórmula, para a confiabilidade é utilizado o valor calculado nas estatísticas do sistema, assim este valor é alterado após um número determinado de execuções, conforme parâmetros ajustados pelo usuário.

5.4 Cenário 3: Ferramenta de *Profiling*

Com o objetivo de demonstrar a reutilização dos componentes implementados, um cenário para a realização dos testes é a utilização do *kernel* para análise de impacto em um aplicativo voltado para a composição dos serviços como uma ferramenta de *profiling*. Neste contexto, foi selecionada a ferramenta desenvolvida que implementa a interface para a composição de serviços utilizando a linguagem inSOA [ZANUZ et al., 2008], ferramenta essa que faz parte do projeto para desenvolvimento do ambiente colaborativo ubíquo do grupo de pesquisa.

A ferramenta para composição, apresentada na Figura 5.11, possui uma interface para o usuário digitar sua composição através da linguagem inSOA e permite as seguintes operações ao usuário:

- Compilar: executa uma análise léxica e semântica da linguagem, gerando um XML para que seja publicado e executado pela *Engine*;
- Publicar: após análise a ferramenta permite que a composição seja publicada no servidor (SOA *Engine*) onde está será executada [ZANUZ, FILIPPETTO et al., 2008];

- Abrir Arquivo: permite que seja carregado um *script* salvo anteriormente na linguagem inSOA.

Figura 5.11 – Interface Ferramenta inSOA

A ferramenta original foi alterada para receber a análise de impacto em suas funcionalidades. Assim foi incluído um novo botão na interface que execute a análise. Por se tratar de um cenário para demonstrar a reutilização dos componentes da ferramenta gImpact, e ter como objetivo a análise sobre composições que estão em fase de desenvolvimento, esta nova funcionalidade não deve onerar a ferramenta já desenvolvida, por este motivo que o projeto da ferramenta gImpact dividiu em dois componentes as regras de negócio e a manipulação da base de dados, permitindo assim que este seja separado e utilizado apenas partes de sua funcionalidade.

Neste cenário para utilização em uma ferramenta de *profiling* será utilizada apenas a funcionalidade que realiza a busca dá árvore de composições e o cálculo do impacto, não utilizando a base de dados para armazenar o histórico ou preferências do usuário. Contudo a fórmula proposta se altera de forma automática, uma vez que não serão consideradas as estatísticas do serviço e preferências do usuário. A Figura 5.12 apresenta a tela após a customização realizada, onde foi incluído um novo botão para realizar o cálculo que por sua

vez é apresentado em uma janela de mensagens, o custo e o total de nodos que fazem parte da árvore de composição.

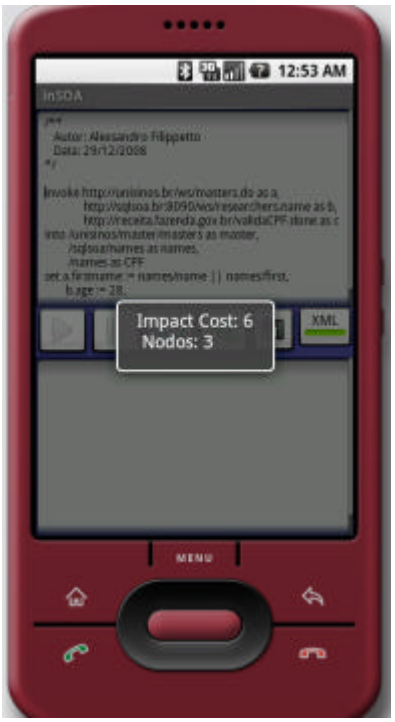

Figura 5.12 – Interface Ferramenta inSOA com *Profiling*

Para esta customização foi utilizado apenas o componente “gImpactControl” apresentado na Figura 4.12, que é acionado através do botão que foi incluído na interface. Este componente realiza uma busca de toda a árvore de composições a partir da composição informada e executa o cálculo de impacto para a composição proposta. A fórmula original, de acordo com os dados que ela recebe, pode se adaptar. Neste trabalho por não ser armazenada as execuções dos serviços o próprio componente ajusta a fórmula, que é apresentada abaixo:

$$I(S) = \sum_{S_i \in S} \left(D(S_i) + L(S_i)\right)$$

Nesta adaptação da fórmula que a ferramenta realiza, serão consideradas a distância de cada serviço (D) e sua localização (L), as demais variáveis são abstraídas da fórmula de modo automático.

Após a customização da ferramenta para a inclusão da funcionalidade de cálculo de impacto, é demonstrada sua utilização através de um cenário de composição. Onde foram

descritas composições e para cada nova implementação antes de sua publicação é realizado uma análise sobre a mesma com a nova implementação.

5.4.1 Estudo de Caso: Consulta de Livros

Para validação das alterações realizadas na ferramenta inSOA, visando acrescentar a funcionalidade de *profiling*. Foi utilizado o estudo de caso apresentado no cenário 1, na seção 5.2, que realiza uma consulta de livros e pesquisa dados sobre o possível comprador no SPC (Serviço de Proteção ao Crédito) e SERASA.

Dentro deste cenário foi descrita na ferramenta a composição para o serviço "getBooks" e a partir da composição original, apresentada na Figura 5.2, foram elaboradas outras duas composições com variações da primeira, visando otimizar sua implementação e execução.

5.4.2 Resultado dos Testes

Após a configuração do ambiente para execução foram realizadas execuções para consultar o impacto e o total de nodos da árvore de composições obtidas através da funcionalidade de *profiling* adicionada a ferramenta original. Estes dados são apresentados na Tabela 5.5.

Após a execução do cálculo para a composição original, procurou-se otimizar seu código através da redução de serviços utilizados ou através de composições que agregassem mais serviços. Uma descrição das alterações realizadas nas composições para cada execução é apresentada a seguir:

- Execução 1: Composição original apresentada no primeiro cenário dos estudos de caso, acrescida de 3 serviços para consulta a SERASA, em um total de 23 serviços;
- Execução 2: Para a segunda execução, foram abstraídos da composição os serviços que realizavam a soma dos valores do livro e adicionando mais serviços em uma única composição. Esta composição é formada por 18 serviços;
- Execução 3: Nesta execução, procurou-se otimizar a composição adicionando mais serviços em uma mesma composição não removendo nenhum serviço que estivesse nas folhas da árvore, assim foram removidas apenas algumas composições que foram incorporadas por outra, ficando um total de 13 serviços.

Tabela 5.5 – Resultado de Avaliação das Composições

	Execução 1	Execução 2	Execução 3
Impacto	82	64	38
Nodos	23	18	13

Durante as execuções o total de nodos da composição sofreu alterações e a localização dos serviços, uma vez que na terceira execução o código foi alterado para que as composições existentes agregassem um maior número de composições, assim influenciando no resultado calculado. A variação do valor do impacto é apresentada no gráfico da Figura 5.13.

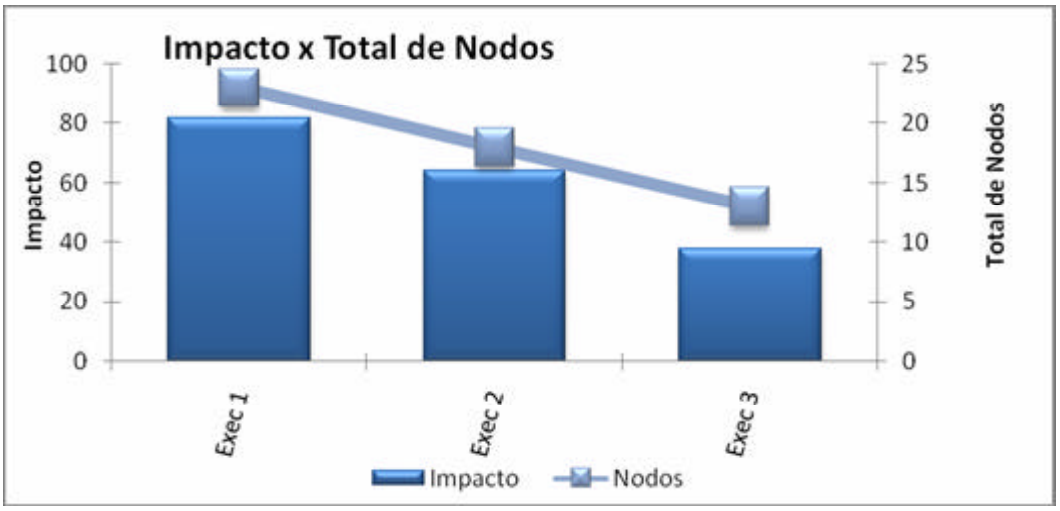

Figura 5.13 – Gráfico Impacto x Total de Nodos

Neste gráfico é possível visualizar que a medida que o número de serviços acessados é reduzido, diminui o impacto calculado. Porém na terceira execução, mesmo se mantendo o mesmo número de serviços reduzidos (um total de cinco) na composição, conseguiu-se uma redução no valor do impacto maior ao se agregar mais serviços em uma mesma composição, tornando assim a árvore gerada com menos níveis de profundidade.

6 CONCLUSÕES

Esta dissertação apresentou um dos principais problemas que condiz com o aumento na utilização de serviços: a otimização para melhor uso das composições [HP SOA, 2008]. Neste sentido o trabalho apresenta uma revisão bibliográfica de tecnologias relacionadas a serviços, como *web services*, BPEL e SOA. Além de plataformas de desenvolvimento voltadas a dispositivos móveis (Android, JME e .NET). A partir destes estudos então é construída uma ferramenta que visa fornecer informações para permitir uma análise das composições visando sua otimização.

Apesar de a arquitetura orientada a serviços estar em diversas pesquisas, o estudo de seu uso em dispositivos móveis ainda são restritos. Com isto, a possibilidade de utilizar esta arquitetura para a computação ubíqua se torna uma área interessante de pesquisa, uma vez que normalmente tais dispositivos possuem uma capacidade limitada de processamento e armazenamento, por este fato a utilização de serviços pode se tornar uma opção para processamentos mais complexos.

Após a concepção da ferramenta para análise de impacto, foram realizados estudos de caso, onde os dois primeiros cenários apresentados demonstram a utilização da ferramenta gImpact para obter informações sobre as composições utilizadas, informações como o impacto/complexidade, calculados através de uma fórmula proposta, o total de serviços que compõe a árvore e médias calculadas de acordo com o número de execuções dos serviços. A variação no custo calculado demonstra a adaptação da fórmula de acordo com as configurações de preferências do usuário e a volatilidade dos serviços disponibilizados na Internet. Desta forma é possível através de uma análise do histórico de execuções e dos valores calculados uma otimização ou a seleção de uma melhor composição para atendimento a um determinado requisito de negócio.

O terceiro cenário elaborado nos estudos de caso, apresenta a utilização de funcionalidades do projeto para compor uma ferramenta de *profiling*, onde foi possível demonstrar o baixo acoplamento de suas funcionalidades, o que permite que partes do *kernel* sejam reutilizados de acordo com o sistema para qual se está modelando. Da mesma forma, a fórmula utilizada possui um determinado grau de adaptação, onde esta se utiliza das variáveis disponíveis no ambiente, como no caso apresentado foram utilizadas a distância dos serviços e sua localização. O que permite que seja avaliada a complexidade das composições visando otimizar seu desenvolvimento.

Com o aumento da complexidade dos serviços é essencial a validação e otimização em sua utilização [HP SOA, 2008]. Neste cenário, através da análise dos estudos de caso é possível identificar a importância das informações fornecidas pelo trabalho para permitir um melhor gerenciamento sobre as composições elaboradas. Possibilitando assim a utilização mais otimizada dos recursos, seja ele de programação ou de equipamentos para execução dos serviços.

6.1 Contribuições

O término da construção da ferramenta gImpact permite elencar as contribuições proporcionadas por ela. As principais delas são destacadas a seguir:

- Ferramenta esperada pelo projeto U-SOA;
- Comunicação através de padrões da indústria baseados em XML;
- Implementação de uma ferramenta que permita seu acesso através de ambientes móveis;
- Elaboração de uma fórmula que permita estimar o impacto da execução de composições;
- Análise sobre serviços permitindo um melhor gerenciamento visando otimizações em sua execução;
- Gerenciamento quanto a alterações realizadas nas composições referentes ao número de serviços que a compõe;
- Informações referentes a confiabilidade dos serviços;
- Histórico de todas as consultas realizadas;
- Estatísticas sobre os serviços consultados através da ferramenta gImpact.

6.2 Limitações

Algumas limitações do trabalho podem ser citadas, sendo que poderão ser solucionadas em uma segunda abordagem:

- Implementação de comunicação real entre celulares baseados na plataforma Android;

- Construção de uma arquitetura conforme apresentada em um ambiente real;
- Permitir que os parâmetros de entrada dos serviços para execução sejam genéricos, permitindo assim a contabilização de tempo para todos os serviços independentes dos parâmetros de entrada.

6.3 Trabalhos Futuros

Espera-se que, com base neste trabalho apresentado, sobre a Análise de Impacto venham a surgir novas pesquisas relacionadas a esta área, o que contribui para o desenvolvimento e composição de serviços. Para trabalhos futuros sugerem-se algumas novas pesquisas, como: criação de grupos ou comunidades para acesso aos celulares, para que com isto seja possível uma maior segurança e controle sobre os recursos dos dispositivos. Outra questão importante seria a extensão da fórmula proposta para utilização de outras variáveis que possam interferir ou gerar um maior custo na execução de serviços.

Outra área a ser explorada é quanto à execução dos serviços, para avaliação de tempo em sua utilização. A ferramenta gImpact, executa as composições para medir o tempo, porém a entrada já deve estar fixa na ferramenta ou a composição deve ser executada sem parâmetros. Desta forma sugere-se a inclusão de um novo componente que adapte uma interface de entrada para os dados de acordo com os parâmetros da composição, tornando dinâmica esta entrada de dados.

7 REFERÊNCIAS

[AHMED e UMRYSH, 2002] AHMED, Khawar Zaman; UMRYSH, Cary E. **Desenvolvendo Aplicações em Java com J2EE e UML**. 1. ed. Rio de Janeiro, Brasil. Ed. Ciência Moderna, 2002. 302p.

[AMAZON WS, 2008] **Amazon Web Services**. Disponível em: <http://soap.amazon.com/

schemas2/AmazonWebServices.wsdl>. Acesso em: 20 dezembro 2008.

[ANDROID, 2008] **Android – An Open Handset Alliance Project**. Disponível em: <http://code.google.com/android>. Acesso em: 28 abril 2008.

[BIH, 2006] BIH, J. **Service Oriented Architecture (SOA): A New Paradigm to Implement Dynamic E-business Solutions.** Ubiquity Volume 7, Issue 30, 2006.

[BRIAND et al., 2006] BRIAND, L. C.; LABICHE, Y.; O'SULLIVAN, L.; SÓWKA, M. M. **Automated impact analysis of UML models**, In: Journal of Systems and Software. Ed. Elsevier, 2006.

[BOOCH et. al., 2005] BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. **UML Guia do Usuário**. Rio de Janeiro, Brasil. Ed. Canpus, 2005. 474p.

[BUSCHMANN et. al., 1996] BUSCHMANN, Frank; MEUNIER, Regine; ROHNERT, Hans; SOMMERLAD, Peter; STAL, Michael. **Pattern – Oriented Software Architecture - A System of Patterns**. Inglaterra. Ed. John Wiley & Sons, 1996. 457p.

[CRESPO, 2000] CRESPO, Sérgio Crespo Coelho da Silva Pinto. **Composição em WebFrameworks**. Tese de Doutorado. Pontifica Universidade Católica do Rio de Janeiro, 2000.

[ECLIPSE, 2008] **Eclipse – An Open Development Plataform**. Disponível em: <http://www.eclipse.org/>. Acesso em: 28 abril 2008.

[FALKL, 2005] FALKL, J. **Service Oriented Architecture Compliance: Initial steps in a longer journey**. Disponível em: <http://download.boulder.ibm.com/ibmdl/pub/ software/dw/webservices/soa-compliance.pdf>. Acesso em: 28 abril 2008. Publicação Dezembro, 2005.

[FOWLER e SCOTT, 2000] FOWLER, Martin; SCOTT, Kendall. **UML Essencial.** 2. ed. Porto Alegre: Ed. Bookman, 2000. 169p.

[FURLAN, 1998] FURLAN, José Davi. **Modelagem de Objetos através da UML.** 1. ed. São Paulo, Brasil. Ed. Makron Books, 1998. 329p.

[GARCIA et al., 2004] GARCIA, Vinicius Cardoso; LUCRÉDIO, Daniel; PRADO, Antonio Francisco do; ALVARO, Alexandre; ALMEIDA, Eduardo Santana de. **Using Reengineering and Aspect-based Techniques to Retrieve Knowledge Embedded in Object-Oriented Legacy System**. In the IEEE International Conference on Information Reuse and Integration (IEEE IRI-2004). Las Vegas, Nevada, USA, 2004.

[HARRISON e TAYLOR, 2006] HARRISON, Andrew; TAYLOR, Ian. **Service-Oriented Middleware for Hybrid Environments.** In: ADPUC – Advanced Data Processing in Ubiquitous Computing. Melbourne, Australia. 2006.

[HIGH, KINDER e GRAHAM, 2005] HIGH Jr, Rob; KINDER, Stephen; GRAHAM, Steve. **IBM's SOA Foundation: An Architectural Introduction and Overview**. Disponível em: <http://download.boulder.ibm.com/ibmdl/pub/software/dw/webservices/ ws- soa-whitepaper.pdf>. Acesso em: 28 abril 2008. Publicação Novembro, 2005.

[HP SOA, 2008] **HP SOA Quality - Deliver outstanding SOA web service performance**. Disponível em: <http://h71028.www7.hp.com/enterprise/cache/484284-0-0-225-121.html>. Acesso em: 10 agosto 2008.

[JACOBSON et al., 1999] JACOBSON, Ivar; BOOCH, Grady; RUMBAUGH, James. **The Unified Software Development Process**, Reading, MA.: Addison-Wesley, 1999. 512p.

[JAVA, 2008] **Java Technology**. Disponível em: <http://www.java.sun.com>. Acesso em: 30 dezembro 2008.

[JDEV, 2008] **Oracle JDeveloper**. Disponível em: <http://www.oracle.com/technology/ products/jdev/index.html>. Acesso em: 11 maio 2008.

[JME, 2008] **Java ME Technology**. Disponível em: <http://java.sun.com/javame/ technology/index.jsp>. Acesso em: 01 maio 2008.

[KOSCIANSKI e SOARES, 2006] KOSCIANSKI, André; SOARES, Michel dos Santos. **Qualidade de Software**. 1. ed. São Paulo, Brasil. Ed. Novatec, 2006. 395p.

[kSOAP, 2008] **kSOAP**. Disponível em: <http://ksoap2.sourceforge.net/>. Acesso em: 02 maio 2008.

[kXML, 2008] **kXML**. Disponível em: <http://kxml.sourceforge.net/>. Acesso em: 02 maio 2008.

[LARMAN, 2004] LARMAN, Craig. **Utilizando UML e Padrões: Uma Introdução à Análise e ao Projeto Orientados a Objetos e ao Processo Unificado**, 2.ed., Porto Alegre: Bookman, 2004. 696p.

[METSKER, 2004] METSKER, Steven John. **Padrões de Projeto em Java**. 1. ed. Porto Alegre, Brasil. Ed. Bookman, 2004. 407p.

[MSDN CF, 2008] **Visão Geral do .Net Compact Framework**. Disponível em: <http://www.microsoft.com/brasil/msdn/Tecnologias/netframework/framework_visaogera.aspx>. Acesso em: 01 maio 2008.

[MSDN FRAMEWORK, 2008] **.NET Framework**. Disponível em: <http://msdn.microsoft.com/pt-br/netframework/default.aspx>. Acesso em: 30 dezembro 2008.

[NETBEANS, 2008] **NetBeans IDE**. Disponível em: <http://www.netbeans.org/>. Acesso em: 11 maio 2008.

[OHA, 2008] **Open Handset Alliance**. Disponível em: <http://www.openhandsetalliance.com/>. Acesso em: 28 abril 2008.

[PIJANOWSKI, 2007] PIJANOWSKI, K. **Visibility and Control in a Service-Oriented Architecture.** Disponível em: <http://msdn2.microsoft.com/en-us/architecture/ bb507204.aspx>. Acesso em: novembro 2007.

[PREE, 1995] PREE, W. **Design Patterns for Object-Oriented Software Development**. 1. ed.: Ed. Addison-Wesley, 1995. 268p.

[RICHTER, 2005] RICHTER, Jeffrey. **Programação Aplicada com Microsoft .NET Framework**. 1. ed. Porto Alegre, Brasil. Ed. Bookman, 2005. 563p.

[SAMPAIO, 2006] SAMPAIO, Cleuton. **SOA e Web Services em Java**. 1. ed. Rio de Janeiro, Brasil. Ed. Brasport, 2006. 151p.

[SÁNCHEZ-NIELSEN et al., 2006] SÁNCHEZ-NIELSEN, Elena; MARTÍN-RUIZ, Sandra; RODRIGUEZ-PEDRIANES, Jorge. **An Open and Dynamical Service Oriented Architecture for Supporting Mobile Services.** In: ICWE - International Conference on Web Engineering. Palo Alto, EUA. 2006.

[SATYANARAYANAN, 2001] SATYANARAYANAN, M. **Pervasive Computing: Vision and Challenges.** In: IEEE Personal Communications. Pág. 10-17. 2001.

[SHIL et al., 2006] SHIL, Assia Ben; AHMED, Mohamed Ben. **Additional Functionalities to SOAP, WSDL and UDDI for a Better Web Services Administration**. In: ICTTA – Information and Communication Technologies. 2006.

[SOAP, 2008] **SOAP Version 1.2 (W3C)**. Disponível em: <http://www.w3.org/TR/2007/REC-soap12-part0-20070427/>. Acesso em: 03 maio 2008.

[SOMMERVILLE, 2004] SOMMERVILLE, I. **Software Engineering**. 7. ed. EUA. Ed. Addison-Wesley, 2004. 784p.

[SQLite, 2008] **SQLite**. Disponível em: <http://www.sqlite.org/>. Acesso em: 08 outubro 2008.

[SU et al., 2004] SU, Xiaoyong; PRABHU, B. S.; CHU, Chi-Cheng; GADH, Rajit. **Middleware for Multimedia Mobile Collaborative System**. In: WTS – Wireless Telecomunications Symposium. California, EUA. 2004.

[THOMAS et al., 2003] THOMAS, Johnson P; THOMAS, Mathews; GHINEA, George. **Modeling of Web Services Flow**. In: CEC – E-Commerce . 2003.

[WEERAWARANA e CURBERA, 2002] WEERAWARANA, Sanjiva; CURBERA, Francisco. **Business Process with BPEL4WS: Understanding BPEL4WS**. Disponível em: <http://www.ibm.com/developerworks/webservices/library/ws-bpelcol1/>. Acesso em: 03 maio 2008. Publicação agosto, 2002.

[WEISER, 1991] WEISER, Mark. **The Computer for the 21st Century**. Sci. Amer., Setembro, 1991.

[WILSON e KESSELMANN, 2000] WILSON, Steve; KESSELMANN, Jeff. **Java Plataform Performance: Strategies and Tatics**. 1. ed. EUA. Ed. Addison-Wesley, 2000. 256p.

[XIAO e ZOU, 2007] XIAO, Hua; ZOU, Ying. **Supporting Change Impact Analysis for Service Oriented Business Applications**, In: International Workshop on Systems Development in SOA Environments. 2006.

[ZANUZ et al., 2008] ZANUZ, L. ; BARCELOS, G. ; FILIPPETTO, A. ; PINTO, S. C. C. S. **U-SOA - Towards a Ubiquitous Platform Based on Service - Oriented Architecture**. In: XIV Simpósio Brasileiro de Sistemas Multimídia e Web (WebMedia), Vila Velha, 2008.

[ZANUZ, FILIPPETTO et al., 2008] ZANUZ, L.; FILIPPETTO, A.; BARCELOS, G. ; PINTO, S. C. C. S. **SOA Engine: Services Compositions Execution in Ubiquitous Environments**. In: XIV Simpósio Brasileiro de Sistemas Multimídia e Web (WebMedia), Vila Velha, 2008.

[ZHANG et al., 2007] ZHANG, Yue; YU, Tao; LIN, Kwei-Jay. **Efficient Algorithms for Web Services Selection with End-to-End QoS Constraints**, In: ACM Transactions on the Web (TWEB). New York, EUA. 2007.

[ZHOU e NIEMELA, 2006] ZHOU, Jiehan; NIEMELA, Eila. **Toward Semantic QoS aware Web Services: Issues, Related Studies and Experience**, In: International Conference on Web Intelligence. 2006.

ANEXO A

Diagrama de Classes Completo (gImpact):