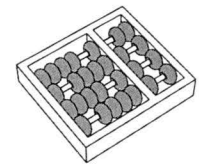



Rômulo José Franco

# "FlexMonitorWS: uma solução para monitoração de serviços Web com foco em atributos de QoS"

ERRATA

Em págs. i, iii, iv, v e vii:
**Onde se lê**: Rômulo José Franco
**Leia-se**: Romulo Jose Franco

Prof. Paulo Lício de Geus
Coord. de Pós-Graduação
Instituto de Computação - Unicamp
Matrícula 10.326-8

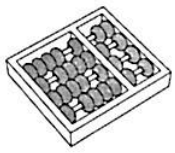
Este exemplar corresponde à redação final da Tese/Dissertação devidamente corrigida e defendida por: Rômulo José Franco

e aprovada pela Banca Examinadora.
Campinas, ____ de _____ de ____

COORDENADOR DE PÓS-GRADUAÇÃO
CPG-IC

Prof. Dr. Paulo Licio de Geus
Coord. de Pós-Graduação
Instituto de Computação - Unicamp
Matrícula 10.326-8

CAMPINAS
2014

Universidade Estadual de Campinas
Instituto de Computação

**Rômulo José Franco**

# "FlexMonitorWS: uma solução para monitoração de serviços Web com foco em atributos de QoS"

Orientador(a): **Prof. Dr. Cecília Mary Fischer Rubira**

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Computação da Universidade Estadual de Campinas para obtenção do título de Mestre em Ciência da Computação.

Este exemplar corresponde à versão final da Dissertação defendida por Rômulo José Franco, sob orientação de Prof. Dr. Cecília Mary Fischer Rubira.

Cecília Mary Fischer Rubira
Assinatura do Orientador(a)

CAMPINAS
2014

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Maria Fabiana Bezerra Muller - CRB 8/6162

F848f

Franco, Rômulo José, 1980-
FlexMonitorWS : uma solução de monitoração de serviços Web com foco em atributos de QoS / Rômulo José Franco. – Campinas, SP : [s.n.], 2014.

Orientador: Cecília Mary Fischer Rubira.
Dissertação (mestrado) – Universidade Estadual de Campinas, Instituto de Computação.

1. Serviços Web - Monitoração. 2. Serviços Web. 3. Garantia de qualidade - Software. 4. Linhas de produto de software. 5. Engenharia de software. 6. Controle de qualidade. I. Rubira, Cecília Mary Fischer,1964-. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Informações para Biblioteca Digital

**Título em outro idioma:** FlexMonitorWS : a solution for monitoring Web services with a focus on QoS attributes
**Palavras-chave em inglês:**
Web services - Monitoring
Web services
Quality assurance - Software
Software product lines
Software engineering
Quality control
**Área de concentração:** Ciência da Computação
**Titulação:** Mestre em Ciência da Computação
**Banca examinadora:**
Cecília Mary Fischer Rubira [Orientador]
Marco Vieira
Eliane Martins
**Data de defesa:** 01-08-2014
**Programa de Pós-Graduação:** Ciência da Computação

# TERMO DE APROVAÇÃO

Defesa de Dissertação de Mestrado em Ciência da Computação, apresentada pelo(a) Mestrando(a) **Rômulo José Franco**, aprovado(a) em **01 de agosto de 2014**, pela Banca examinadora composta pelos Professores Doutores:

**Prof(a). Dr(a). Marco Vieira**
**Titular**

**Prof(a). Dr(a). Eliane Martins**
**Titular**

**Prof(a). Dr(a). Cecília Mary Fischer Rubira**
**Presidente**

Instituto de Computação
Universidade Estadual de Campinas

# FlexMonitorWS: uma solução para monitoração de serviços Web com foco em atributos de QoS

**Rômulo José Franco**

01 de agosto de 2014

**Banca Examinadora:**

- Prof. Dr. Cecília Mary Fischer Rubira (*Orientador*)
- Profa. Dra. Eliane Martins
  Instituto de Computação - Universidade Estadual de Campinas
- Prof. Dr. Marco Vieira
  Departamento de Engenharia Informática - Universidade de Coimbra
- Profa. Dra. Ariadne Carvalho
  Instituto de Computação - Universidade Estadual de Campinas (*Suplente*)
- Prof. Dr. Patrick Brito
  Instituto de Computação - Universidade Federal de Alagoas (*Suplente*)

# Abstract

Web services are used as a way of obtaining a Service Oriented Architecture (SOA). With interoperable, dynamic and distributed aspects such services add business values are software units with high cohesion and are used to integrate business applications. In a SOA context, service providers must offer guarantees of the servces operations. This warranty is carried out by attributes Quality of Service (QoS) type contracts entered into SLA (Service Level Agreement). QoS attributes can have fluctuations or changes of state over time, given that a service operates in an environment of high dynamics and high unpredictability inherent in the SOA context properties. Given this scenario, there is a clear need to understand the fluctuations in the QoS attributes. Therefore, it is essential to apply a monitoring which allows to know the QoS attributes values to understand the overall context of the environment that operates the service. A good monitoring solution must offer flexible ways to monitor different QoS attributes (e.g. performance, availability and reliability) in different ways to operate, considering different targets linked to the service (e.g. server, network and server application).

By analyzing existing solutions through a Systematic Literature Review identified that solutions do not support the flexibility in monitoring. Against this background, this thesis proposed a solution FlexMonitorWS monitoring of Web services and IT infrastructure resources connected to the Web services. FlexMonitorWS adopts techniques from Software Product Lines to create a monitors family from the existing variability in the Web services monitoring systems. Three case studies were performed to assess the tool feasibility, obtaining satisfactory results in delivering QoS attributes values and understanding to environment that operates the Web service. In the end, conclusions, contributions and directions for future work are presented.

# Resumo

Serviços Web são usados como uma das formas de se obter uma Arquitetura Orientada a Serviços (SOA). Com aspectos interoperáveis, dinâmicos e distribuídos, tais serviços agregam valores de negócio, são unidades de software com alta coesão e são utilizados para integração entre aplicações empresariais. Em um contexto de SOA, provedores de serviços devem oferecer garantias de funcionamento de seus serviços. Esta garantia é realizada através de atributos de *Quality of Service* (QoS) inseridos em contratos do tipo SLA (*Service Level Agreement*). Atributos de QoS podem ter flutuações ou mudança de estado ao longo do tempo, dado que um serviço opera em um ambiente de alta dinamicidade e alta imprevisibilidade que são propriedades inerentes ao contexto SOA. Diante deste cenário, há uma clara necessidade de se conhecer as variações que ocorrem nos atributos de QoS. Para isso, é fundamental aplicar uma monitoração que possibilite conhecer os valores de atributos de QoS para compreender o contexto geral do ambiente que opera o serviço. Uma boa solução de monitoração deve oferecer meios flexíveis de monitorar diferentes atributos de QoS (e.g. disponibilidade, desempenho e confiabilidade), de diferentes modos de operar, considerando diferentes alvos ligados ao serviço (e.g. servidor, rede e aplicação servidora). Por meio de uma Revisão Sistemática da Literatura identificamos que as soluções encontradas não apoiam a flexibilidade na monitoração. Face a este contexto, esta dissertação propôs a FlexMonitorWS uma solução de monitoração de serviços Web e de recursos de infraestrutura de TI ligada ao serviço Web. A FlexMonitorWS adota técnicas de Linhas de Produtos de Software para criar uma família de monitores a partir da variabilidade de software existente em sistemas de monitoração de serviços Web. Três estudos de caso foram executados para avaliar a viabilidade da ferramenta, obtendo-se resultados satisfatórios na entrega de valores de atributos de QoS e na compreensão do ambiente que opera o serviço Web. Ao final, apresentamos conclusões, contribuições e direções para trabalhos futuros.

# Agradecimentos

Agradeço a Deus acima de tudo, e a minha família pelo apoio nos momentos difíceis. No coração de cada um de vocês encontro refúgio. Minha maior alegria sempre foi reencontrar vocês, minha gratidão e meu amor.

Agradeço a minha orientadora Professora Cecília Rubira, por ter me orientado de forma compreensível e seus ensinamentos serão levados por toda a minha vida.

Agradeço aos amigos que fiz no Instituto de Computação e que contribuíram tanto com o projeto quanto na minha formação durante esta etapa. Em especial, a minha querida amiga Amanda Nascimento. Também agradeço pela participação de Patrick Brito e Leonardo Tizzei.

Agradeço aos profissionais, funcionários, professores do Instituto de Computação da UNICAMP, sempre me auxiliaram com muita atenção nas etapas até a finalização do projeto.

Meus sinceros agradecimentos aos meus amigos que tanto amo, e que me apoiaram sempre de forma incondicional, Evandro Cézar, Marcos Abraão, Mariana Aguiar, Fábio Junio e Marisa Padilha.

Agradeço também a empresa KNBS que deu oportunidade de iniciar este projeto em paralelo com os trabalhos. Lá pude fazer grandes amizades, José Ricardo Navas e Priscila Correa.

Finalmente, agradeço a todos que de alguma maneira contribuíram para que pudesse subir neste pódio.

*"Agradeço todas as dificuldades que enfrentei; não fosse por elas, eu não teria saído do lugar. As facilidades nos impedem de caminhar. Mesmo as críticas nos auxiliam muito."*

Francisco Candido Xavier

# Sumário

# Lista de Tabelas

# Lista de Figuras

# Capítulo 1

# Introdução

Serviços têm sido amplamente usados para uma infinidade de aplicações, principalmente no contexto do modelo de Arquiteturas Orientadas a Serviços. Este modelo faz uso da interoperabilidade, importante propriedade de serviços, especificamente serviços Web. A interoperabilidade é definida como a capacidade de vários sistemas heterogêneos compartilharem informação de maneira eficiente e é um requisito fundamental [Geraci91]. A Arquitetura Orientada a Serviços (SOA) é uma das principais abordagens que permite aumentar o grau de interoperabilidade entre sistemas. Dado que SOA interrelaciona diferente serviços utilizando padrões de interfaces e protocolos já consolidados neste cenário. Serviços são independentes de plataformas e linguagens de implementação [Papazoglou07]. Logo, a computação orientada a serviços torna o reúso interorganizacional um forte impulsionador de negócios utilizando serviços.

Assim, serviços são projetados sob a perspectiva fundamental da heterogeneidade de sistemas que interagem entre si por meio de composição de serviços [Papazoglou06]. Esta composição de serviços reflete a capacidade de combinar atividades de maneira uniforme em uma ordem predefinida, dada uma regra de negócio [Papazoglou07]. Serviços Web são de maneira geral a principal tecnologia deste contexto SOA.

Em resumo, dada as características de SOA como a heterogeneidade, interoperabilidade e alta dinamicidade, em que os serviços são removidos e criados em tempo de execução, contribui com um ambiente complexo de gestão, principalmente em manter a qualidade do serviço. A satisfação do requisito não funcional garante a Qualidade do Serviço (do inglês *Quality of Service*) prestado. Dada a disseminação de serviços, muitas vezes são exigidos critérios. Tais critérios podem ser garantidos contratualmente por meio de Acordos de Nível de Serviço (SLA - do inglês *Service Level Agreement*).

Neste contexto de SOA e qualidade do serviço, é i) comum a existência de vários clientes com diferentes requisitos e possivelmente requisitos conflitantes; ii) flutuações nos valores de atributos de qualidade de serviços (QoS) são recorrentes iii) manutenções nos sistemas e no ambiente acarretam na degradação dos atributos de QoS; iv) o ambiente de execução contém instabilidades periódicas e contribui também com a degradação do

atributo de QoS.

A adoção de métodos como Linhas de Produtos de Software (LPS) está interrelacionada com modelos como SOA, uma vez que ambos apóiam a reutilização de software. A visão original de LPS é formada por um domínio concebido a partir das necessidades do negócio, formulado como uma família de produtos e com um processo de produção associado [Campbell08]. Este domínio apóia a construção rápida e evolução de produtos customizados conforme mudanças nos requisitos dos clientes. Variabilidade de Software é um conceito fundamental em LPS e refere-se a capacidade de um sistema de software ou artefato ser modificado, customizado ou configurado, para ser utilizado em um contexto específico [Gurp01]. Consequentemente, ao se projetar variabilidades de software em LPS, decisões de projetos são postergadas e resolvidas em fases posteriores do desenvolvimento [Svahnberg05].

As variabilidades podem ser inicialmente identificadas por meio de características. Característica é uma propriedade de sistema que é relevante para alguma parte interessada e é usada para capturar partes comuns ou variáveis entre sistemas de uma mesma família [Chang07]. Outro artefato importante da LPS é a Arquitetura de Linhas de Produtos (ALP). Uma ALP provê uma visão global na qual são identificadas as partes variáveis e comuns de uma LPS em termos de elementos arquiteturais e suas configurações.

## 1.1 Contexto do problema

Serviços estão presentes em nosso cotidiano. Um bom exemplo é o comércio eletrônico. Supondo que um cliente efetue uma compra na internet e nessa compra ocorra i) a validação do número do cartão de crédito, ii) a efetivação do pagamento do pedido junto a operadora de cartão, e entre outras etapas ocorre iii) o despacho do produto via empresa de transporte terceirizada. Todas estas etapas são exemplos que compõem funcionalidades que possivelmente são executadas por meio da composição de serviços. Nota-se nesta composição a integração de negócios tendo diferentes organizações participantes, cada uma desempenhando seu papel. Pressupondo da existência de requisitos não funcionais da operação de cada serviço, endereçados através de atributos de QoS, a primeira operação, por exemplo, deveria estar sempre disponível. Enquanto a segunda operação, deveria ser realizada com alta confiabilidade uma vez que constitui uma operação bancária, e ainda incluir alto desempenho e disponibilidade. A terceira, integridade, garantindo a correta conclusão da operação.

Uma organização que fornece um serviço para a composição, deve garantir que seu serviço tenha atributos de QoS, tais como, alta disponibilidade, alto desempenho, acurácia, robustez, integridade e confiabilidade dos seus serviços. A degradação dos atributos de QoS pode acarretar a não satisfação dos acordos firmados (SLA) e a consequente penalidade financeira ou legal. Assim, a organização pode oferecer um serviço composto, demais serviços que não estão sob sua alçada, mas que estão na composição podem ter

seus atributos degradados também. Por isso, é imprescindível que a organização compreenda de onde surge a degradação, qual a organização responsável e se for ela, em quais recursos ocorreu a degradação (e.g. recursos de infraestrutura de TI, como, rede, servidor, container e o próprio serviço).

Portanto, as organizações que disponibilizam serviços, precisam assegurar a qualidade do ambiente como um todo colaborando entre si em uma cooperação mútua e coletiva entre elas. Disso decorre na melhoria contínua da qualidade do serviço oferecido ao consumidor final. De maneira geral, ferramentas de monitoração apoiam a qualidade, posto que atributos de QoS são conhecidos durante a monitoração. Um simples exemplo da aplicação da monitoração seria, na presença da degradação de atributos de QoS, um possível envio de notificações aos interessados.

## 1.2 Definição do Problema

Embora exista uma demanda crescente por monitoração de atributos de QoS, projetar e implementar ferramentas para tal atividade é uma tarefa não trivial. Devido ao caráter específico de cada atributo de QoS em relação a sua forma de monitoração, as soluções existentes tem abordagens pontuais e monitoram um ou dois atributos de QoS [Moser08, Wang09, Ivanovic10, Katsaros11 e Muller12]. Além disso, normalmente as abordagens consideram um número limitado de alvos, sendo a maioria exclusiva somente sobre serviços. Conclui-se que há uma carência de soluções para monitoração que apoie e gerencie diferentes funcionalidades e diferentes alvos, conforme requisitos dos usuários e que facilite a evolução e extensão de funcionalidades existentes.

Quanto às diferentes funcionalidades apoiadas, temos, por exemplo, serviços são monitorados ativa ou passivamente. Por um lado, no modo ativo, testes são utilizados a fim de invocar serviços periodicamente para medir seus valores de QoS [Tian04]. Estes testes podem executar serviços paralelamente ou de modo sequencial. Por outro lado, no modo passivo, a ferramenta é responsável por interceptar requisições de clientes aos serviços a fim de medir atributos de qualidade [Artaiam08, Thio05]. Diferentes atributos de QoS podem ser monitorados (e.g. disponibilidade, confiabilidade, tempo de resposta, segurança e integridade de serviços) [Souza11]. Conforme o subconjunto de atributos monitorados, pode ser necessário invocar serviços ou somente verificar se estes estão disponíveis (e.g. utilizando ping ou latência que é um utilitário adotado para testar conectividade entre equipamentos) [Schoene09]. A monitoração de atributos de QoS pode ser intermitente ou contínua [Godse10]. Além disso, a monitoração pode ser em outros recursos ligados ao serviço, como recursos de infraestrutura de TI. Estes recursos constituem diferentes alvos da monitoração. Assim, a monitoração sobre tais recursos apóia na identificação da degradação de atributos e na compreensão de fatores ligados a atributos de QoS. Adicionalmente, algumas das funcionalidades da monitoração são mutuamente exclusivas. Por exemplo, caso a monitoração seja passiva, não é possível estabelecer um intervalo de

monitoração contínua visto que a medição de QoS só será realizada quando existir uma requisição a ser interceptada [Godse10].

Diferentes clientes podem ter diferentes requisitos referentes às funcionalidades que a ferramenta de monitoração deve apoiar. Tais requisitos podem sofrer modificações no decorrer do tempo, inclusive em tempo de execução, dado a dinamicidade inerente a sistemas orientados a serviços e a diversidade de usuários que tais sistemas apoiam [Papazoglou07].

De um lado, há um conjunto de necessidades e mudanças de requisitos que determinam o quão uma solução de monitoração deveria atender incluindo os diferentes atributos de QoS, diferentes alvos, diferentes modos de operação. E por outro lado, técnicas como *Linhas de Produtos de Software* podem ser usadas para apoiar tais necessidades ao ter em sua base a reutilização de software.

Idealmente, explorar o conceito de LPS utilizando a importante vertente da variabilidade em soluções de monitoração de serviços Web pode contribuir em uma solução com apelo a flexibilidade, tão importante em sistemas desta natureza. Neste sentido, este trabalho almeja responder a seguinte questão de pesquisa:

**RQ1**: Como aprimorar a flexibilidade das ferramentas de monitoração de atributos de qualidade de serviços a fim de melhor apoiar diferentes perfis de monitoração?

A fim de responder a tal questão, temos a seguinte hipótese:

- $H_1$: A adoção de técnicas de linhas de produtos para desenvolver uma família de monitores de atributos de QoS e com múltiplos alvos da monitoração, aprimora a flexibilidade de tais ferramentas dada a necessidade em atender diferentes requisitos ligados a ambientes desta natureza.

## 1.3 Visão Geral da Solução Proposta

O foco central deste trabalho é o desenvolvimento de uma família de monitores de atributos de qualidade de serviços (QoS) e de recursos de infraestrutura de TI. A família de monitores gerados pela LPS considera gerar monitores específicos. Cada monitor está associado a um perfil de monitoração, o que permite separar conceitualmente o requisito desejado pelo interessado no monitor da sua materialização em um produto físico e executável.

Diferentemente das soluções encontradas na literatura [Souza11, Muller12, Halima08], cada monitor gerado pela LPS, executa independentemente de plataforma e permite a monitoração simultânea de vários serviços e de vários recursos de TI como alvos, na obtenção de valores de um ou mais atributos de QoS. A avaliação de cada atributo de QoS considera tanto eventos temporais, como permite a inspeção do estado e condição de recursos de infraestrutura de TI e também, se existem políticas de segurança implementadas no serviço Web. Vulnerabilidades em serviços Web podem ser exploradas, uma vez que

utilizamos os conceitos da [WSInject10] para avaliar e obter o atributo de QoS robustez. Desta maneira, a nossa solução também pode injetar falhas em serviços Web e detectar vulnerabilidades nestes.

Para nos apoiar na criação de nossa solução, foram estudadas outras soluções existentes para monitoração de QoS a fim de identificar funcionalidades comuns e variáveis entre elas. Além de identificar lacunas que tais soluções não preenchiam, acrescentamos a nossa solução como possíveis funcionalidades.

Assim, as funcionalidades identificadas foram mapeadas para características e posteriormente documentadas em um modelo de características. Após, as variabilidades identificadas no modelo foram mapeadas e implementadas na arquitetura baseada em componentes da linha de produtos. Esta arquitetura foi decisiva, pois ela expõe a variabilidade arquitetural da solução e reflete a existência de alternativas do projeto arquitetural proposto. Conclui-se portanto, a criação a partir de uma combinação de componentes para gerar uma família de monitores, é factível.

Em consequência, a avaliação da solução foi realizada a partir de estudos de caso baseado em três cenários, cenário controlado, cenário real e um cenário de injeção de falhas para aferir atributos de QoS como robustez. Foi estabelecido nestes cenários um conjunto de funcionalidades que deveriam ser extraídas da LPS em forma de produto (neste caso monitor), para atender a uma demanda do cenário ligado ao estudo de caso. Por fim, a partir dos estudos de caso, foram avaliadas de forma qualitativa e obtivemos a partir da análise, resultados positivos e satisfatórios gerados pela monitoração. A análise nos permitiu concluir também sobre a flexibilidade em monitorar diferentes atributos de QoS, diferentes serviços Web e diferentes recursos ligados ao serviço. Através dos estudos de caso identificamos limitações da nossa solução e projetamos trabalhos futuros.

O restante desta dissertação está organizado da seguinte forma: o capítulo 2 define os conceitos necessários sobre Arquiteturas Orientadas a Serviços e Sistemas de monitoração, e também, serão apresentados Fundamentos de Reuso. No capítulo 3 uma Revisão Sistemática da Literatura foi realizada durante o projeto e serão apresentados os detalhes da revisão. No capítulo 4 descrevemos todos os passos usados para uma proposta e implementação da solução de monitoração em várias fases. No capítulo 5, foram realizados três estudos de caso e serão apresentados detalhadamente sobre as etapas que os compuseram. Por fim, o capítulo 6 apresenta as conclusões da dissertação, e projeta direções que podem ser exploradas em trabalhos futuros.

# Capítulo 2

# SOA, Sistemas de monitoração e Fundamentos sobre reuso

Neste capítulo, serão apresentados os principais conceitos sobre Arquiteturas Orientadas a Serviços e Sistemas de monitoração. A primeira parte do capítulo visa relacionar a definição de Arquitetura Orientada a Serviços com Serviços Web, uma das formas de se obter estes tipos de arquiteturas. A segunda parte do capítulo trata sobre Sistemas de monitoração típico de serviços Web.

Durante a fundamentação teórica sobre tipos de sistemas de monitoração de serviços Web, é oferecida neste capítulo uma taxonomia com a finalidade de identificar de forma hierárquica as diferentes categorias e subcategorias correspondentes a estes sistemas.

Serão apresentados também, os conceitos sobre os principais métodos e modelos adotados para a criação de uma LPS e sobre reusabilidade, arquiteturas de software e plataformas utilizadas para codificação.

## 2.1 Arquitetura Orientada a Serviços (SOA)

A *Arquitetura Orientada a Serviços* (SOA) é um estilo arquitetural emergente para desenvolver e integrar aplicações empresariais, em que funcionalidades são agrupadas em processos de negócios e empacotadas como serviços interoperáveis [Papazoglou06]. Serviços podem ser definidos como módulos de software que provê funcionalidades de negócios e são independentes do estado ou contexto de outros serviços [Papazoglou07]. Composição de serviços é um conceito importante em computação orientada a serviços, visto que permite que funcionalidades de serviços existentes sejam combinadas a fim de oferecem um serviço composto com maior valor agregado. Além disso, os serviços são publicados através de interfaces simples, permitindo que eles sejam encontrados e invocados por outras aplicações clientes. A utilização de padrões para cada uma destas atividades (publicar, encontrar e invocar) contribui para o sucesso de SOA ao apoiar a comunicação entre

aplicações escritas em diferentes linguagens de programação e executando em diferentes plataformas [Dustdar05].

A Figura 2.1 mostra os principais elementos da Arquitetura Orientada a Serviços. Num primeiro momento, o serviço é *publicado* em repositório de serviços ou *broker* de serviços, que permite que os serviços sejam encontrados (passo 1) . Os consumidores de serviços acessam o *broker* de serviços para *encontrar* os serviços que necessitam (passo 2) e, caso encontrem, *invocam* o serviço acessando-o diretamente (passo 3).

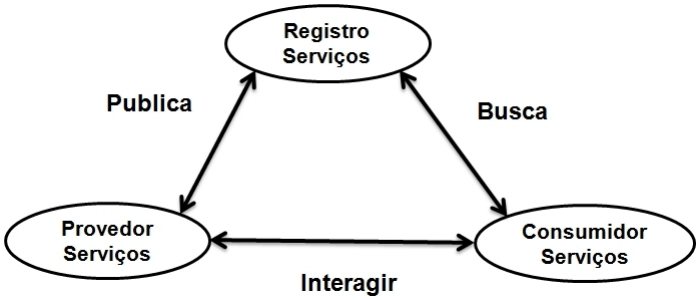



Figura 2.1: Arquitetura Orientada a Serviços (Adaptado de Huhns et. al. 2005)

As tecnologias baseadas em serviços Web podem ser utilizadas para se construir sistemas orientados a serviços no contexto da Internet [Alonso04]. Serviços Web podem ser definidos *como um sistema de software projetado para apoiar interações máquina-máquina em uma rede, que contém uma interface descrita em um formato processável pela máquina (WSDL - Web Services Description Language), e com o qual outros sistemas interagem-se orientados por sua descrição e usando mensagens SOAP - Simple Object Access Protocol (tipicamente sobre HTTP - HyperText Transfer Protocol) serializadas usando XML - eXtensible Markup Language [W3C03]. UDDI (Universal Description, Discovery and Integration)* [Alonso04] é uma especificação que define serviços Web e modelos de dados para suporte à descrição e descoberta de outros serviços Web.

A Figura 2.2 mostra a total aderência de serviços Web ao modelo proposto pela Arquitetura Orientada a Serviço. Note na Figura 2.2, os mesmos elementos presentes no modelo são implementados pelos recursos diretamente ligados a serviços Web com protocolo SOAP.

Existem outros tipos de serviços que usam protocolos diferentes. Recentemente serviços REST (*Representation State Transfer*) tem sido amplamente usado em alternativa a serviços Web que usam o protocolo SOAP. Neste trabalho, focamos em serviços Web protocolo SOAP.

Os serviços Web podem ser simples ou compostos. A **composição de serviços** Web pode ser realizada através de linguagens de especificação de processos como BPEL (*Business Process Execution Language*) [Liu10] ou mesmo a partir de linguagens diversas como Java, C++ e PHP. Por exemplo, um serviço composto poderia ser criado como um *script* PHP que invoca de forma programática outros serviços Web. Neste cenário, caso

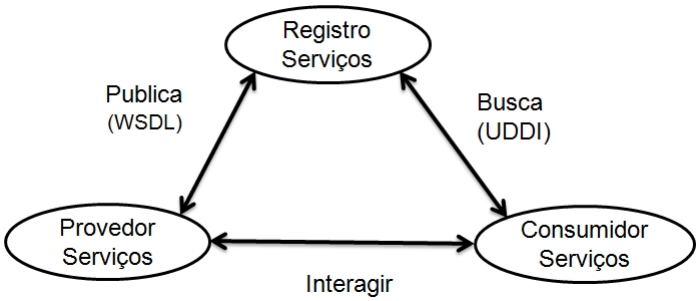



Figura 2.2: Arquitetura Orientada a Serviços Web (Adaptado de Huhns et. al. 2005)

a composição seja realizada do lado do provedor, o *script* PHP é exposto como Serviço Web.

Quando esta composição de serviços é realizada, a manutenção do ambiente como um todo para atender a qualidade do serviço prestado previamente estabelecida usa de mais esforços para manter esta qualidade. A verificação da composição dos serviços é realizada mediante a aplicação de sistemas de monitoração.

## 2.2 Sistemas de monitoração de serviços Web

Sistemas de monitoração de serviços Web, tem sido explorado na academia e na indústria em atendimento a diferentes objetivos. Para ampliar nosso entendimento sobre estes sistemas, foi necessário a junção dos conceitos, objetivos, formas e fatores para compor uma definição com mais acurácia que culminou em uma taxonomia. Para esta finalidade, recorremos a Engenharia de Software Experimental a partir de uma Revisão Sistemática da Literatura seguindo as diretrizes proposta por Kitchenham (2004) e pode ser vista no Capítulo 3. Esta revisão contribuiu de forma efetiva no entendimento de sistemas de monitoração de Serviços Web.

Tipicamente, a monitoração sobre serviços Web pode ser realizada por meio da interceptação de mensagens e testes de uso do serviço efetuando invocações sobre ele e obter respostas para posterior análise. Podem ser usadas sondas (do inglês *Probes*) inseridas em pontos estratégicos. Quando alguma anomalia é detectada ocorre a notificação dos interessados na monitoração.

Para realizar este tipo de monitoração questões básicas de projeto devem ser consideradas: 1) Onde ocorrerá a monitoração (alvo)? 2) O que se deseja obter com a monitoração ou o que deveria ser acompanhado (atributos de QoS)? 3) De que modo isso pode ser realizado (modo de operação)? 4) Qual a frequência da monitoração? 5) Quais os meios de se obter resultados ou alertas gerados pela monitoração (notificação)?

Estas questões podem ser mapeadas em uma classificação para monitoração de serviços Web e é descrita a seguir.

### 2.2.1 Taxonomia de monitoração de serviços Web

Considerando Serviços Web, a monitoração pode ter alvos, atributos de QoS, modos de operação, frequência da monitoração e modos de notificação, agrupados da seguinte forma:

1. **Alvos da monitoração**

A Figura 2.3 apresenta os possíveis alvos da monitoração, sendo o 1) Serviço, implantado em uma 2) Aplicação Servidora que está instalada em um 3) Servidor físico, em que a comunicação entre provedor e consumidor ocorre pela 4) Rede.

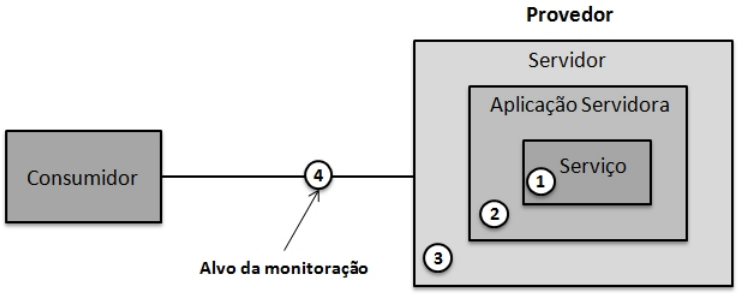



Figura 2.3: Possíveis alvos da monitoração (adaptado de H. Ludwig. 2003)

Quando o alvo é serviço, a monitoração pode ser realizada analisando as mensagens que são trocadas entre provedor e consumidor. A Aplicação Servidora é um container, servidor ou uma plataforma que disponibiliza um ambiente para a instalação de serviços (*e.g. Apache Tomcat*[1]). O foco da análise da Aplicação Servidora é a partir de Logs gerados por ela que determinam o estado e condição dos serviços que foram implantados internamente à Aplicação Servidora.

O Servidor é um recurso físico que faz parte dos recursos computacionais e que tem uma Aplicação Servidora instalada. O Servidor físico é passível de monitoração dos recursos que estão dentro dele, como unidades de disco, memória, entre outros recursos.

Por fim, a Rede é um ponto ou um endereço IP estabelecido na monitoração que é foco da comunicação durante o uso do serviço. Monitorar roteadores, *gateways* pode ser uma saída no entendimento do contexto de onde emergem problemas de disponibilidade e desempenho.

Ao monitorar nos quatro alvos é considerada uma monitoração no **contexto geral** por onde podem ocorrer desvios que geram impactos e degradam atributos de QoS.

2. **Atributos de QoS**

[1] Aplicação Servidora - *Apache Tomcat - http://tomcat.apache.org/*

O objetivo primário da monitoração é a coleta de informações relacionadas a atributos de QoS. Posteriormente, ao aplicar a monitoração, os resultados provenientes desta, podem ser usados para fins específicos e serão vistos na Seção 2.2.3. São considerados neste caso, os atributos de QoS separados por alvo conforme Tabela 2.1.

Tabela 2.1: Atributos de QoS por alvo de monitoração

| **Atributo de QoS** | **Serviço** | **Servidor** | **Aplicação Servidora** | **Rede** |
|---|---|---|---|---|
| Desempenho | • | | | • |
| Disponibilidade | • | | | • |
| Confiabilidade | • | | | |
| Acurácia | • | | | |
| Robustez | • | | | |
| Condição do hardware | | • | | |
| Falhas detectadas em Logs | | | • | |

Os atributos de QoS relacionados ao serviço, são atributos contidos no modelo recomendado pela W3C [W3C03]. Demais recursos monitorados como no servidor e arquivos de log da Aplicação Servidora apoiam a compreensão de fatores ligados a degradação de atributos de QoS [Ludwig03].

3. **Modo de operação**

Para obter valores de atributos de QoS é necessário realizar algum tipo de teste com o serviço ou obter informações durante a sua execução. Há duas maneiras elencadas: Interceptação e Invocação. Na **Interceptação** [Delgado04, Schroeder95], a monitoração ocorre ao esperar que o serviço seja usado para obter informações sobre ele a partir das mensagens que são trocadas entre provedor e consumidor. Na **invocação** testes reais são aplicados a partir de requisições geradas para o serviço e, por conseguinte uma análise da resposta é realizada [Wetzstein09]. Para recursos como Servidor e Aplicação Servidora o modo de **Inspeção** pode ser aplicado, como por exemplo, para analisar arquivos de Log gerados pelo Aplicação Servidora, ou inspecionar o estado e condição do hardware do servidor. Pode ser considerado o mesmo que a invocação, ao emitir comandos com protocolo ICMP (e.g. comando *ping*) [Ludwig03], porém de forma assíncrona.

4. **Frequência de execução**

A monitoração pode ser **contínua** quando ela ocorre o tempo todo em um intervalo de tempo muito pequeno como em milissegundos. **Intermitente**, quando o monitor pode atuar de forma periódica, coletando somente em tempos previstos**.** A monitoração pode ser realizada somente quando o serviço for acionado por outra aplicação que não seja a monitoração (e.g. uso normal do serviço). Neste caso, a monitoração somente seria executada quando novas requisições/respostas ao serviço fossem emitidas. O que torna a monitoração não proativa e tendo valores de atributos de QoS desatualizados.

5. **Notificação**

Para o acompanhamento dos resultados, ao final do processo de monitoração, os resultados deveriam ser enviados ou mantidos em disco para o interessado na monitoração. Esta atividade pode ser realizada por meio do envio de mensagens ou via gravação de arquivo de Log.

### 2.2.2 Perfis de monitoração

A Figura 2.4, a taxonomia contém diversos elementos agrupados em suas devidas categorias, a combinação destes elementos gera um monitor que atende a um **Perfil de monitoração**. Desta forma, vários monitores podem ser criados atendendo diferentes perfis de monitoração.

Existem diferentes objetivos que a monitoração apoia, dada as necessidades distintas de provedores e consumidores. Um perfil de monitoração, portanto, é uma combinação proveniente da taxonomia ligada a um monitor para atender a um objetivo específico conforme é descrito alguns destes objetivos da monitoração na Seção 2.2.3.

Note ainda na Figura 2.4, os elementos que compõem a **Definição do alvo da monitoração**, quando todos são agrupados em um único perfil de monitoração, oferece um perfil de monitoração no **Contexto geral**, em que todos os alvos possíveis da taxonomia serão monitorados.

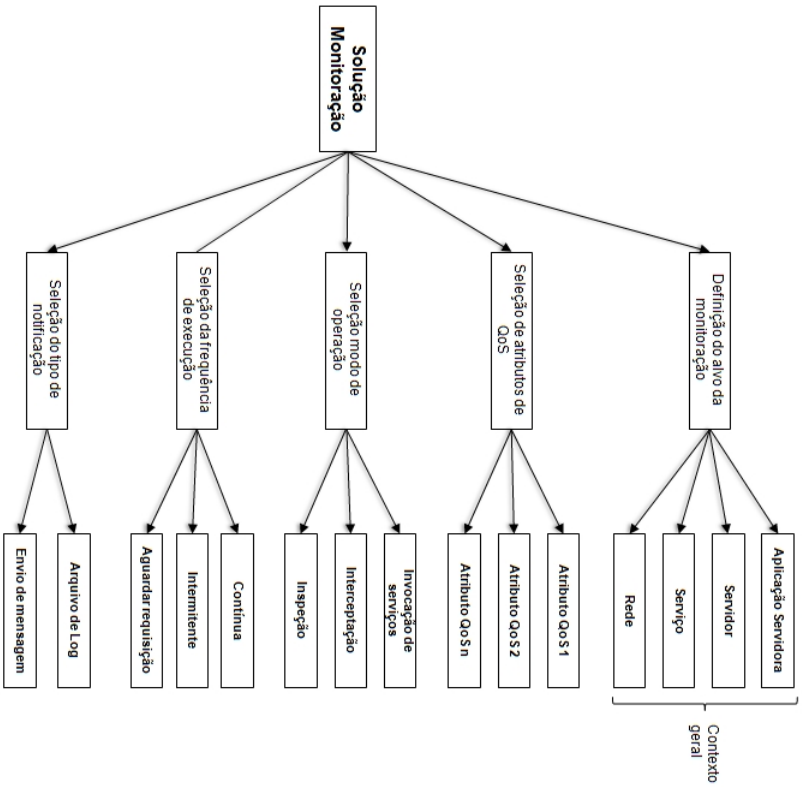



Figura 2.4: Taxonomia de sistemas de monitoração de serviços Web e Recursos de TI

### 2.2.3 Objetivos da monitoração

A monitoração de serviços Web visa diferentes objetivos, como exemplos têm-se os seguintes itens:

1. Melhoria do processo de seleção e descoberta de serviços (*Broker*);
2. Técnicas de *self-healing*[2] , adaptação dinâmica e recuperação dinâmica;
3. Detecção de violação em acordos SLA (*Service Level Agreement*).

No primeiro item, a melhoria do processo de seleção tenta aprimorar o funcionamento do UDDI, que não fornece meios de efetuar buscas baseado em atributos de QoS [Thio05]. Como consequência, não é possível a diferenciação entre serviços com operações para a mesma finalidade por atributos de QoS [Tian04]. O consumidor escolhe um serviço de forma aleatória, não sabendo se a sua escolha atende as suas expectativas. Para melhoria e aperfeiçoamento deste processo de seleção do serviço é empregado um módulo semelhante a um *broker* ou uma extensão do UDDI que fica entre o provedor e o consumidor [Thio05, Tian04].

[2]self-healing - auto-cura, tradução literal

De acordo com a Figura 2.5, o *broker* é alimentado por provedores a fim de manter seus níveis de QoS atualizados (passo 1). Quando o consumidor efetua uma invocação ao serviço (passo 2), primeiramente esta ocorre entre o consumidor e o *broker* que efetua uma escolha do serviço relativo ao QoS que melhor atenda o consumidor (passo 3 ao passo 7), e retorna o serviço escolhido e posteriormente a ligação ocorre entre consumidor e provedor (passo 8) [Weider07].

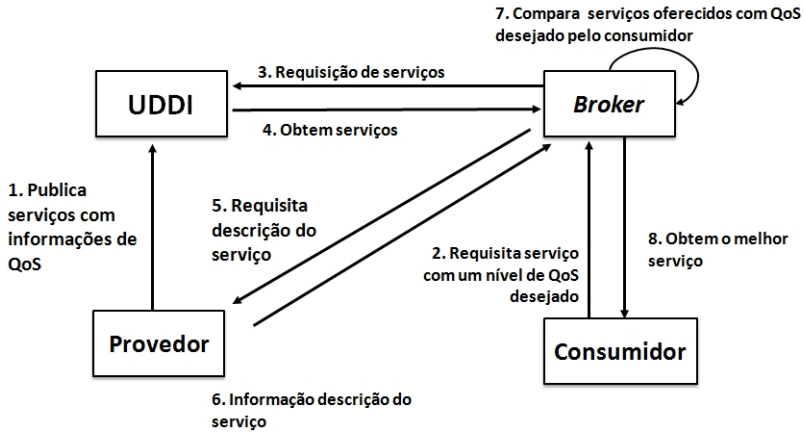



Figura 2.5: Visão geral do funcionamento de um *broker* (adapatado de Weider D. Yu et. al. 2007)

No item 2, as técnicas de *self-healing*, adaptação dinâmica e recuperação dinâmica, são comuns envolvendo atributos de QoS como disponibilidade, escalabilidade, capacidade e confiança (do inglês *reliability*). As técnicas são aplicadas quando quaisquer destes atributos não correspondem a um nível esperado. São técnicas complexas como, por exemplo, estratégias de balanceamento de carga (do inglês *load balance*) que permitem alocar serviços em servidores redundantes quando o desempenho e disponibilidade estão degradados [Gmach08].

Por fim, em relação ao item 3, a monitoração pelo lado do provedor se espera oferecer níveis adequados de atributos de QoS e permitir o gerenciamento do ambiente onde o serviço está implantado. Obviamente, há interesses do lado do provedor em aumentar o número de consumidores do seu serviço, quando QoS são atendidos ganhando fidelidade [Marzolla10]. Pelo lado do consumidor, visa uma abordagem mais fiscalizadora na detecção de violação de atributos de QoS acordados em SLA. A monitoração neste caso promove redução de custos ao permitir maximizar atributos de QoS como *reliability* e *throughput* [Marzolla10, Al-Masri09]. Quando algum atributo é violado, alarmes e notificações são disparados para que consumidores e provedores tomem conhecimento. Além desta notificação, pode acarretar em penalidades e preços ao provedor [Thio05].

Além dos itens acima mencionados, tem-se a composição de serviços, em que o provedor pode desempenhar um papel de coordenação ao agregar em um único serviço, a junção de outros serviços que não estão sob seu controle [Dustdar05]. A composição de serviços é

realizada através de serviços de terceiros. Cada serviço de terceiro pode representar uma necessidade específica, ou um contrato SLA com uma finalidade específica também para cada serviço.

Além disso, dependendo do tipo de composição de serviço, para o provedor as suas dependências são resolvidas em nível de comunicação fim-a-fim (ponto-a-ponto). Porém, para o consumidor que efetua uso de um serviço composto, pode representar meramente uma caixa preta. Se a monitoração ocorre deste ponto (no caso a partir do consumidor com alvo no provedor cujo serviço é composto), é mais complexo determinar em qual provedor necessariamente ocorre uma degradação no atributo de QoS.

Os objetivos da monitoração descritos aqui usam de atributo de QoS para avaliar a qualidade do serviço prestado. Um atributo de QoS pode representar um ou mais requisitos não funcionais. Um modelo de atributos de QoS deve ser considerado para representar as diferentes formas de cada atributo e as diferentes perspectivas vistas de cada lado da comunicação, seja do provedor como também do consumidor.

## 2.3 Modelo de Atributos de QoS aplicado a serviços Web

Há na literatura diferentes modelos de atributos de QoS endereçados como requisitos para serviços Web [Truong06, Cabrera12, Marzolla10 e Ran03]. Alguns autores destacam padrões de qualidade e normas como ISO/IEC 9126-1 [Cabrera12 e Iso01], outros classificam atributos mediante uma taxonomia [Truong06, Marzolla10 e Ran03].

No trabalho de S. Araban et. al. (2004) os atributos são categorizados como fatores de QoS separando-os em atributos por domínios internos e externos a partir da perspectiva do provedor. Menascé (2002) e Marzolla et. al. (2010) efetuam uma separação por pontos de vista representando os interesses da avaliação de atributos de QoS do lado do provedor e consumidor.

Além dos modelos acima mencionados, a W3C (*World Wide Web Consortium*) [W3C03] faz uma importante recomendação a partir de um conjunto de atributos de QoS. Embora a descrição do funcionamento de cada atributo seja resumida, a maioria dos atributos está contida na relação de atributos da ISO/IEC 9126-1, bem como nos modelos analisados por outros autores [Truong06, Cabrera12, Marzolla10 e Ran03].

A adoção do modelo de atributos de QoS oferecido pela W3C é a escolha do modelo para validar este projeto, uma vez que a W3C é o órgão que rege as regras na Web. E por ser um conjunto de atributos de QoS pode-se obter os níveis em ambas as partes da comunicação entre consumidor e provedor de Serviços Web.

Há duas maneiras de coletar informações sobre um dado atributo de QoS elencadas neste trabalho:

1. Calcular a partir de **eventos temporais** como tempo inicial da requisição contra

tempo final da resposta a uma invocação;

2. Verificar internamente ao Serviço, Servidor ou arquivo de Log a existência de alguma falha, determinando o **Estado e Condição** ou a existência do atributo de QoS internamente ao serviço através de padrões e políticas [Ran03].

Os atributos de QoS contemplados nesta primeira versão da solução proposta são os descritos a seguir.

### 2.3.1 Disponibilidade (do inglês *Availability*)

O serviço web deve estar disponível para consumo imediato. Essa disponibilidade é a probabilidade de que o sistema está instalado e operante em um determinado espaço de tempo. Este espaço de tempo ou janela de tempo é uma escolha aleatória a ser considerada na monitoração por tempos previstos de execução. A janela de tempo escolhida na monitoração se for contínua irá garantir que o nível deste atributo estará sempre atualizado, mas pode acarretar em degradação do desempenho, posto que o serviço sempre será alvo de testes, concorrendo assim com invocações de usos rotineiros do serviço. Ao passo que, se for intermitente com longos períodos de inatividade da monitoração, o nível ao ser consultado pode oferecer um valor desatualizado. Este atributo pode ser obtido a partir da Equação 2.1 [W3C03, Gunther98 e Ran03]:

$$disp = 1 - \frac{uptime}{totaltime} = \frac{uptime}{((uptime) + (downtime))} \tag{2.1}$$

Onde:
*Disp* é a variável que armazena o resultante da fórmula;
*upTime* é o tempo total que o serviço está disponível durante a janela de tempo da monitoração;
*downTime* é o tempo total que o serviço está indisponível durante a janela de tempo da monitoração.

O valor resultante é em unidade percentual do tempo de monitoração que o serviço está disponível. Exemplo: o serviço Web está disponível 80

### 2.3.2 Desempenho (do inglês *Performance*)

O desempenho de um serviço Web representa o quão rápido uma solicitação de serviço pode ser concluída. Ela pode ser medida em termos de rendimento, o tempo de resposta, a latência, o tempo de execução, e o tempo de transação, e assim por diante [W3C03]. O rendimento é dado a partir do número de solicitações ao serviço Web respondidas em um determinado intervalo de tempo. O tempo de resposta é o tempo necessário para completar uma solicitação ao serviço Web. Latência é o atraso de ida e volta entre o envio

de um pedido e ao receber a resposta. O tempo de execução é o tempo gasto por um serviço Web para processar a sua sequência de atividades. Finalmente, o tempo de transação representa o tempo ocorrido enquanto o serviço Web está completando uma transação. Desta vez transação pode depender da definição de transação de serviço [Ran03, W3C03]. Aqui utilizamos o tempo de resposta do serviço para obter o nível deste atributo de QoS. A Equação 2.2 utilizada é [Gunther98]:

$$desem = \frac{\sum_{n}^{i} (responseTime - requestTime)^{i}}{totalSuccessRequests} \tag{2.2}$$

Onde:
*Desem* é a variável resultante da fórmula;
*responseTime*, é o tempo do disparo da requisição;
*requestTime*, é o tempo ao receber uma resposta de uma determinada requisição;
*totalSuccessRequests*, é o total das requisições que obtiveram sucesso.

A Equação 2.2 soma todos os tempos de resposta baseado em todas as requisições disparadas que obtiveram sucesso (com resposta) e obtém a média a partir do número total de requisições com sucesso. O resultado é expresso em milissegundos. Exemplo: o tempo de resposta do serviço Web é de 240ms.

### 2.3.3 Confiabilidade (do inglês *Reliability*)

Serviços Web devem ser fornecidos com alta confiabilidade. Confiabilidade aqui representa a capacidade de um serviço Web em executar suas funções requeridas sob condições estabelecidas por um intervalo de tempo [W3C03]. A confiabilidade é a medida geral de um serviço Web em manter a sua qualidade de serviço mesmo em condições inóspitas [Ran03]. A medida geral de um serviço Web está relacionada com o número de falhas por dia, semana, mês ou ano. A confiabilidade também está relacionada com a entrega garantida e a correta ordem das mensagens que estão sendo transmitidas e recebidas por consumidores de serviços e provedores de serviços. No entanto, a obtenção deste atributo pela solução de monitoração não garante eventuais falhas internas no processamento a uma requisição. A confiabilidade é relacionada a entrega da mensagem que será efetivamente entregue de acordo com a política estabelecida.

Dado a complexidade em mensurar este nível do atributo de QoS, baseamos em um trabalho que utiliza o padrão *WS-Reliability* (ou *Web Service Reliable Messaging* – WS-RM) para assegurar a existência do atributo de QoS de confiabilidade no serviço [Rosenberg09].

A Figura 2.6 apresenta internamente no XML como é identificado a política WS-*Reliability* (WS-RM – *Reliable Messaging*).

Quando existe a aplicação deste padrão inserido na pilha do protocolo SOAP, há uma garantia de entrega da mensagem ao destinatário, bem como a correta ordem do envio das mensagens será mantida.

```
targetNamespace="http://unicamp.ic.br/" name="NewWebService">
▼<wsp:Policy xmlns:wsrmp="http://docs.oasis-open.org/ws-
 rx/wsrmp/200702" wsu:Id="NewWebServicePortBindingPolicy">
  ▼<wsrmp:RMAssertion>
     <wsp:Policy/>
   </wsrmp:RMAssertion>
   <wsam:Addressing wsp:Optional="true"/>
  </wsp:Policy>
```

Figura 2.6: Representação parcial - XML da política *WS-Reliability* (WS-RM – *Reliable Messaging*)

Para se obter o nível deste atributo de QoS baseou-se na existência do padrão embutido no protocolo SOAP.

### 2.3.4 Acurácia (do inglês *Accuracy*)

Serviços Web devem ser fornecidos com alta precisão. Precisão aqui é definida como a taxa de erro gerado pelo serviço Web. O número de erros que o serviço gera em mais de um intervalo de tempo deve ser minimizada [Ran03, Gunther98, W3C03].

A Equação 2.3 apresenta como pode-se obter o nível do atributo de QoS [Gunther98]:

$$acur = 1 - \frac{nFaults}{totalRequests} \tag{2.3}$$

Onde,
*Acur*, é a variável resultante;
*nFaults*, é o número de falhas obtidas durante a janela de tempo de monitoração;
*totalRquests*, é o número de requisições disparadas na janela de tempo de monitoração.

A Equação 2.3 obtém o número de falhas a partir do total de requisições realizadas. Se durante a monitoração ocorreu um número de requisições não respondidas ou que geraram falhas é contabilizado e posteriormente no final, os valores são computados.

O nível do atributo de QoS é expresso em percentual. Sendo a taxa de acurácia o próprio nível. Exemplo: a taxa de acurácia do serviço Web é 90% durante o tempo de monitoração.

### 2.3.5 Robustez (do inglês *Robustness*)

Serviços Web devem ser fornecidos com alta robustez. Robustez aqui representa o grau em que um serviço Web pode funcionar corretamente, mesmo na presença de entradas inválidas, incompletas ou contraditórias [W3C03]. Serviços Web ainda devem funcionar mesmo que parâmetros incompletos sejam fornecidos para o pedido de invocação de serviços [Ran03].

Existem inúmeras técnicas que podem ser consideradas. Segundo Ran (2003) robustez pode ser realizada a partir de técnicas de injeção de falhas para avaliar esse atributo. Salas (2012) aplica estas técnicas juntamente com uma ferramenta WSInject (2010). A WSInject é utilizada com o intuito de detectar vulnerabilidades no sistema. Salas (2012) cita Cachin et. al (2000), a vulnerabilidade de um sistema está ligada a robustez uma vez que é uma falha de um ativo ou controle que pode ser explorada por uma ameaça. Pode ser uma falha maliciosa ou não, introduzida de forma intencional ou acidental durante as fases de desenvolvimento. Uma típica vulnerabilidade em sistemas de Internet é manter portas TCP/IP abertas sem supervisão e que podem ser exploradas [Salas12].

Serviços Web podem ser alvos de diversos ataques uma vez que são sistemas abertos que estão em constante comunicação com outros serviços, onde o canal principal de comunicação é a Internet. Padrões utilizados na criação como XML, HTML sob protocolos como HTTP gera grandes desafios na área de segurança destes sistemas.

O procedimento adotado neste projeto visa a identificação de vulnerabilidades a partir da solução que executará um conjunto de scripts. Esta técnica é aplicada sobre uma carga de trabalho (do inglês - *workload*) (XML de requisição ao serviço válida, operante e sem falhas), e a modificação desta, inserindo falhas nocivas ao servidor por substituição a fim de obter uma carga de falhas (do inglês - *faultload*).

Ao aplicar esta técnica, é possível injetar falhas na requisição ao serviço e aguardar a resposta. Durante a avaliação da resposta, efetua-se uma análise baseada no código de status HTTP (do inglês - *HTTP Status Code*) obtido da resposta à requisição ao serviço Web. Este código determina como a requisição com falha foi tratada do lado do servidor e pode determinar o comportamento do servidor em relação ao ataque. Há três tipos de códigos básicos:

**200 - OK**

Padrão de resposta bem sucedida, completada, porém, não pode-se afirmar se foi aceita ou processada.

**400 - *Bad Request***

A requisição não pode ser cumprida devido a sintaxe ruim. Porém, pode-se afirmar que a resposta é robusta uma vez que o servidor detectou a requisição com falha.

**500 - *Internal Server***

O servidor encontrou uma condição inesperada e o impediu de processar a requisição. Algumas vulnerabilidades podem ser detectadas aqui. Pode-se no retorno deste código, vir juntamente mensagem contendo divulgação de informações que põe em risco o próprio servidor (e.g. informações como nome de usuário, caminhos, nomes de programas usados, linguagens)

Internamente a cada um destes códigos há uma subdivisão com status mais detalhados, mas estes grupos nos permite já identificar algo sobre a vulnerabilidade.

Baseado também em Salas (2012) adotamos dois tipos de ataques descritos por ele para criação de uma carga de falhas. Estes ataques confundem o analisador de XML do lado do servidor e acabam culminando durante o processamento em executar comandos mal intencionados ou acessar dados não autorizados.

1. ***MalFormed XML***

   Inserir fragmentos de XML mal formados, como não fechar tags, adicionar novos elementos, substituir tags do padrão SOAP. Este tipo de ataque pode forçar o servidor a interpretar estes fragmentos e gerar falhas, e estas, podem expor informações relevantes sobre o servidor e tornando-o vulnerável.

2. ***XML Bomb Injection***

   XML é uma linguagem recursiva primitiva e em determinadas situações os analisadores (do inglês *parser*) XML podem expandir documentos muito pequenos em documentos maiores. Este tipo de ataque pode gerar uma carga ao ser expandido do lado do servidor em excesso de tal maneira que o consumo de tempo de processamento e memória pode chegar ao colapso e se esgotar, tornando o serviço indisponível. Ou seja, um tipico ataque de negação de serviço (do inglês - *Denial of Service*).

Há muitos outros tipos de ataques que podem ser explorados por substituição de cadeia de caracteres. Em Salas (2012) pode ser conferida uma relação de diferentes tipos de ataques a serviços Web. Aqui representamos somente um exemplo das inúmeras possibilidades. Além disso, a solução terá uma flexibilidade em permitir adicionar *scripts* a partir de parâmetros. Confira detalhes do funcionamento de injeção de falhas na Seção 4.3.4.

Em resumo, a partir desta estratégia não podemos garantir em níveis a robustez de um serviço Web por ser complexo as formas de ataques e há uma grande quantidade de tipos de ataques, uma vez que depende da criatividade do atacante. Contudo, podemos identificar alguma falha que pode ser explorada e definir se durante uma injeção de falhas foi possível detectar vulnerabilidades. Se no âmbito da realização dos testes, injetados as falhas e identificadas as vulnerabilidades, definimos que o serviço Web não contém robustez. Se durante os testes, o serviço passar por todos, garantimos a neutralidade do teste, sem garantias de que há robustez. Nenhuma vulnerabilidade foi encontrada para o conjunto de testes definido, mas não asseguramos robustez para outros testes que não foram realizados.

Tabela 2.2: Atributos de QoS elencados para monitoração

| Descrição | Tipo | Unidade de medida |
|---|---|---|
| Disponibilidade | Evento temporal | Percentual |
| Desempenho | Evento temporal | Milissegundos |
| Confiabilidade | Estado ou condição | Booleana (V ou F) |
| Acurácia | Evento temporal | Percentual |
| Robustez | Estado ou condição | Booleana (V ou F) |
| Estado e condição do hardware | Estado ou condição | Booleana (V ou F) |
| Falhas em arquivos de log | Estado ou condição | Booleana (V ou F) |

### 2.3.6 Estado e condição do hardware

Segundo Wang et. al. (2009) e Michlmayr et. al. (2009) a monitoração de outros recursos ligados ao serviço é capaz de detectar eventos que podem afetar o comportamento do serviço. A monitoração neste caso avalia todos os eventos e estados relevantes que podem ser monitorados.

Aqui neste caso, optamos por avaliar o estado e condição de recursos como Servidor. No caso do Servidor, o estado de unidades de disco e memória é obtido analisando as oscilações que podem ter algum impacto no comportamento do serviço. Unidades de disco que estão cheias prejudicam a memória como em Sistemas Operacionais Windows onde a paginação é feita diretamente no disco rígido.

### 2.3.7 Falhas em Arquivos de Log

Avaliar arquivos de Log na obtenção de falhas é uma abordagem que administradores de servidores comumente executam manualmente. A análise é realizada obtendo informações diretamente no log, avaliando linhas que são categorizadas com erros com severidade criticas ou alertas. Geralmente, servidores de aplicação como Apache Tomcat utilizam a abordagem de *Logging* baseada em *Apache Log4j*[3]. Cada linha do arquivo de log tem uma categoria pertencente a uma classe crítica de alerta, informativa ou somente visualizável em modo de depuração (e.g. *logging* configurado para alertas - preenchem o arquivo de log com a categoria WARN - <mensagem de alerta>).

A ideia, portanto, é efetuar este procedimento que é realizado de forma manual para uma forma automatizada. Ao definir para a solução de monitoração abrir o arquivo de log e inspecioná-lo obtendo informações criticas e alertas, separando-as e notificando os responsáveis pelas falhas identificadas no arquivo de Log.

A Tabela 2.2 apresenta um resumo dos atributos de QoS elencados para monitoração e os tipos de coletas para obter valores sobre cada um deles.

[3] *Apache Log4j - Apache Logging Services - http://logging.apache.org/*

## 2.4 Fundamentos sobre reuso

Aqui, serão apresentados os conceitos sobre os principais métodos e modelos adotados para a criação de uma LPS e sobre reusabilidade, arquiteturas de software, plataformas utilizadas para codificação e métodos e modelos aplicados para implementação do projeto.

### 2.4.1 Linhas de Produto de Software

Conforme definição de Clements e Northrop (2001), uma Linha de Produto de Software (LPS), ou Família de Sistemas [Sommerville95], é planejada para produzir um conjunto de produtos de software com alto grau de similaridade entre si, que atendam as necessidades específicas de uma missão ou segmento de mercado, e que são desenvolvidas de forma prescritiva a partir de um conjunto de artefatos básicos, chamados ativos centrais[4] ou do núcleo. Os ativos centrais tem que lidar, de maneira sistemática, com as diferenças e semelhanças dos produtos, respectivamente, variabilidades[5] e 'comunalidades'[6], são 'customizadas' de acordo com as necessidades dos produtos individuais ao serem instanciados [Capilla05, Gomaa05 e Sinnema04].

De maneira complementar, Gomaa (2002) define a Engenharia da Linha de Produto de Software (SPLE) como o processo que envolve a análise, projeto e implementação da Linha de Produto de Software de modo a satisfazer os requisitos de todas as aplicações alvo. Linden et al. (2007) ressaltam que a SPLE baseia-se na distinção fundamental de desenvolver para reuso e desenvolver com reuso. Enquanto o desenvolvimento para reuso é obtido através do processo de Engenharia de Domínio, o desenvolvimento com reuso é obtido através do processo de Engenharia de Aplicação. Estes dois processos são paralelos e concorrentes.

### 2.4.2 Gerenciamento de variabilidades de software

Em ambas as fases de Engenharia de Domínio e de Aplicação, gerenciamento de variabilidade de software é um conceito chave. Primeiramente, o termo variabilidade de software, conforme descrito por Sinnema & Deelstra (2007), refere-se às escolhas que permitem engenheiros customizar, alterar, configurar ou estender artefatos da família de produtos para serem utilizados em contextos específicos. Consequentemente, ao projetar variabilidades de software em LPS, decisões de projetos são postergadas e resolvidas em fases posteriores do desenvolvimento, por exemplo, durante o tempo de execução de um produto particular [Svahnberg05]. Essencialmente, isso é possível ao utilizar o gerenciamento de variabilidades, especialmente durante a derivação de produtos.

[4] Ativos centrais do inglês, *Core Assets*
[5] Variabilidade, do inglês *Variability*
[6] Comunalidade, do inglês, *Comunalities*

Estas decisões de projeto postergadas são denominadas de ponto de variação. Um ponto de variação é, então, um local do artefato de software em que uma decisão de projeto pode ser tomada. De maneira complementar, variantes são alternativas de projeto associadas a este ponto [Kim06], ou seja, um valor ou instância que pode 'preencher' um ponto de variação. A associação de um ponto de variação a uma variante é chamado de tempo resolução de variabilidade. O tempo de resolução de variabilidades indica em qual momento uma ou mais variantes serão associadas a um determinado ponto de variação, sendo um fator importante no gerenciamento de variabilidades [Krueger02]. As condições necessárias para que um ponto de variação seja associado a uma variante devem ser explicitamente modeladas.

### 2.4.3 Características e Modelos de Características

As variabilidades podem ser inicialmente identificadas por meio de características. Característica é uma propriedade de sistema que é relevante para alguma parte interessante e é usada para capturar partes comuns ou variáveis entre sistemas de uma mesma família [Chang07]. As características de um sistema de software são documentadas em um modelo de características. Um Modelo de características consiste dos seguintes elementos principais: (i) Diagrama de características, representando a decomposição hierárquica das características; (ii) Regras de composição para as características; e (iii) Análise racional das características.

O diagrama de características representa as características padrão de uma família de sistemas em um domínio e o relacionamento entre elas. Um dos relacionamentos possíveis é o agrupamento de características. Características podem ser obrigatórias, opcionais ou alternativas, sendo que características alternativas só tem sentindo dentro de um agrupamento de características.

A Figura 2.7 representa parte de um modelo de características de um aparelho de microondas. As características Contador regressivo de tempo e Idioma são obrigatórias, enquanto que a característica Luz é opcional. O idioma do aparelho de microondas pode ser neste exemplo, em português ou em inglês, representadas pelas características alternativas Inglês e Português. As características alternativas Inglês e Português são especializações da característica Idioma. Ainda, configuração de características é uma instância válida do modelo de características.

### 2.4.4 Abordagens para Projeto de implantação LPS

O projeto de implantação de uma LPS é a adoção de uma forma de desenvolvimento que determinará como se obter os ativos centrais que irão apoiar nos outros processos da LPS. Esta adoção visa facilitar o entendimento de onde se devem buscar informações para elaboração dos artefatos. Um sistema implementado do zero, tem uma abordagem

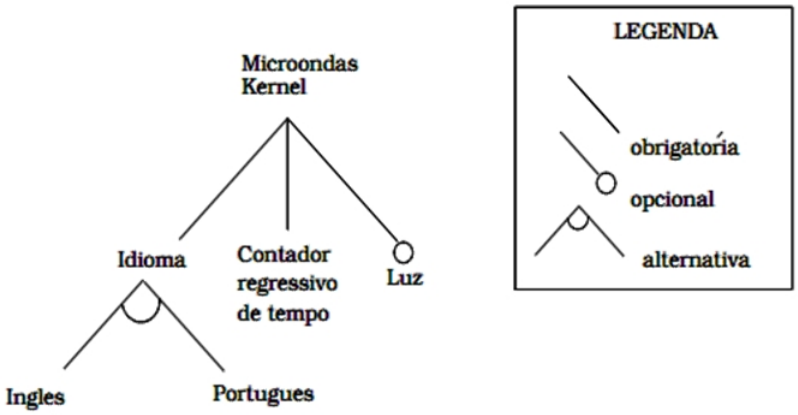



Figura 2.7: Parte de um modelo de características de um Microondas

inicial diferente de sistemas legados a serem transportados para LPS. Ambos podem ter abordagens distintas. O projeto de implantação apoiado na adoção de LPS pode ser abordado de três maneiras [Krueger02a]:

**Proativa:** Nesta abordagem a organização analisa, projeta e implementa uma LPS para apoiar todo o escopo de produtos necessários. A partir da análise e do projeto são implementados os artefatos de software (componentes, arquitetura da linha, definição de processo, definição do escopo e requisitos, entre outros) para a LPS.

**Reativa:** Nesta abordagem a organização cresce sua linha de produtos incrementalmente conforme a demanda de novos produtos ou novos requisitos em produtos existentes. Conforme a necessidade, os artefatos são construídos, utilizados nos produtos já existentes e também alimentando o núcleo de artefatos da LPS. Esta abordagem oferece uma transição para LPS mais rápida e menos dispendiosa do que a proativa.

**Extrativa:** Nesta abordagem a organização tira proveito de sistemas de software existentes, extraindo o código comum e variável para a LPS. Esta abordagem permite a organização adotar a LPS muito rapidamente.

Adotamos abordagem do projeto proativa e para apoiar, foram utilizadas técnicas e diretrizes da Engenharia de LPS a partir do Método PLUS (Seção 2.5.1).

### 2.4.5 Interface de desenvolvimento FeatureIDE

A FeatureIDE [Kastner09] é uma ferramenta baseada em Eclipse[7] que apoia o desenvolvimento de LPS por permitir aplicar as diferentes técnicas apropriadas do tempo de

[7] *Eclipse Foundation – www.eclipse.org*

resolução de variabilidades. As principais técnicas apoiadas pela FeatureIDE são agregação/delegação, herança, parametrização, sobrecarga, reflexão computacional, programação orientada a aspectos entre outros.

A ferramenta possibilita criar o modelo de características e consequentemente aplicar a validação deste modelo. As restrições são criadas de forma visual. Tais restrições representam uma ou mais relações de dependências entre características. Como por exemplo, características mutuamente exclusivas ou inclusivas são impostas no modelo a partir de lógica booleana. A facilidade destas abordagens oferecidas pela ferramenta permite ganhar tempo e visualizar o modelo mais coerente com o que se pretende com a LPS.

## 2.5 Metodologia de LPS

Na literatura existem diferentes metodologias para se construir LPS. São algumas delas, FORM [Kang98], PuLSE [Bayer99], Framework proposto por Pohl (2005) e o *Product Line UML-Based Software Engineering* (PLUS) proposto por Gomaa (2004). O método PLUS se destaca por ser baseado em UML, o que facilita a sua adoção, uma vez que existem diversas ferramentas que apoiam modelos UML. Este método foi utilizado na definição da solução desse presente trabalho e será revisitado nas seções posteriores.

O método PLUS proposto por Gomaa (2004) descreve um ciclo evolucionário para o desenvolvimento de LPSs. Ele estende as notações padrões de UML para dar suporte aos conceitos de variabilidades inerentes a LPSs. O método PLUS pode ser usado de forma abrangente em todas as etapas da LPS. Para tanto, ao considerar o desenvolvimento de uma família de produtos que constituem uma LPS, é necessário substituir as atividades de desenvolvimento de software baseado em modelo para os sistemas individuais de modelagem de requisitos, modelagem de análise e modelagem de projeto (do inglês *design*) por atividades de desenvolvimento que abrangem a família de produtos.

Diferentes modelos produzidos na execução do método como o modelo de casos de uso, modelo de características (ver Seção 2.4.3) e o modelo de classes são relacionados sob o ponto de vista da variabilidade. Ou seja, os elementos variáveis e comuns de cada modelo são relacionados com os elementos de outros modelos. Tabelas são usadas para representar esses relacionamentos, que não são necessariamente de um-para-um. Eventualmente, um elemento de um modelo pode corresponder a mais de um elemento noutro modelo. A rastreabilidade da variabilidade entre os modelos facilita a manutenção e evolução da variabilidade [Berg05].

Estes diferentes modelos ao estender os modelos UML utilizam de estereótipos. Como por exemplo, ao definir no diagrama de características os elementos, eles podem ser: obrigatórios (estereótipo *kernel*), opcionais (estereótipo *optional*) ou alternativos (estereótipo *alternative*). Outros estereótipos também são adotados como [Gomaa04]:

- Zero ou uma do grupo de característica (**zero-or-one-of-feature-group**) onde

dado um grupo de características opcionais ao menos uma deve ser escolhida, neste caso as características pertencentes a este grupo são mutuamente exclusivas.

- Apenas uma de um grupo de características (**exaclty-one-of-feature-group**) semelhante a anterior, por ser a seleção mutuamente exclusiva, contudo, a seleção implica em tornar a característica deste grupo obrigatória em cada produto da linha.

- Pelo menos uma do grupo de característica (**at-least-one-of-feature-group**) a seleção de ao menos uma característica é restritiva a este grupo.

- Zero ou mais do grupo de característica (**zero-or-more-of-feature-group**) aqui não há restrição à seleção da característica deste grupo, ela pode ser ou não incluída no produto.

Aqui neste trabalho, foi utilizado o símbolo de agregação nos diagramas às características pertencentes a um dado grupo, bem como, foi removido dos nomes dos estereótipos de grupos a palavra of-feature-group.

## 2.6 Desenvolvimento baseado em componentes

O desenvolvimento baseado em componentes (DBC) é uma técnica de desenvolvimento de software que se baseia na construção rápida de sistemas a partir de componentes pré-fabricados [Breivold07]. Um componente de software é uma unidade de composição com interfaces e dependências de contexto explicitamente especificadas, que pode ser fornecido isoladamente para integrar sistemas de softwares desenvolvidos por terceiros. Uma interface identifica um ponto de interação entre um componente e o seu ambiente [Baelen00]. Um componente possui interfaces providas, através das quais ele declara os serviços oferecidos ao ambiente e interfaces requeridas, pelas quais ele declara os serviços do ambiente dos quais ele depende para funcionar [Baelen00]. Dois componentes podem ser ligados entre si por um conector que além de intermediar o relacionamento e a comunicação entre componentes pode ser responsável pela implementação de algum atributo de qualidade [Garlan00].

A Figura 2.8 mostra um exemplo da representação de dois componentes com suas interfaces providas e requeridas e um conector ligando ambos. O componente *Microondas* fornece um conector (*connMicroondasLuz*) para que exista um relacionamento com o componente *Luz*.

A combinação de LPS com DBC tem sido aplicada por demostrar ser uma técnica rápida e eficiente ao derivar produtos de um conjunto de artefatos reusáveis [Atkinson02].

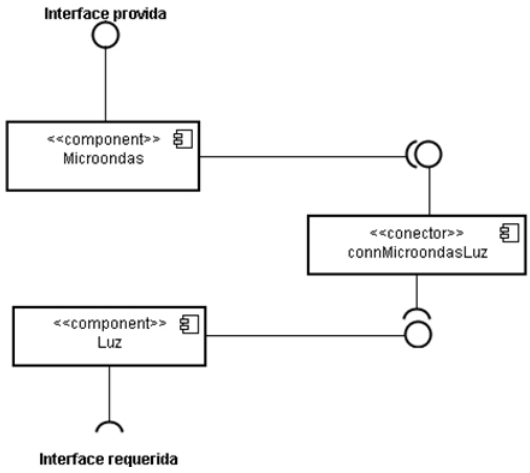



Figura 2.8: Representação de um diagrama de componentes

### Arquitetura da Linha de Produto

Gomaa e Bubak (2008) enfatizam que a Arquitetura da Linha de Produto (ALP) engloba a arquitetura dos membros da linha de produto de software a fim de permitir a derivação das arquiteturas de produtos individuais através da seleção de componentes variantes e opcionais apropriados, ou seja, a arquitetura da linha oferece explicitamente mecanismos de variação que apoiam a família de produtos da LPS.

A Figura 2.8 mostra um exemplo de parte de uma arquitetura de linha de produto baseada em componentes. Neste exemplo, dois componentes arquiteturais, *Microondas* e *Luz* se comunicam por meio do conector arquitetural *connMicroondasLuz*. Cada componente arquitetural representa uma funcionalidade do sistema, enquanto o conector arquitetural conecta os componentes [Garlan00]. Um conector também pode ser responsável por uma parcela dos atributos de qualidade do sistema, tais como distribuição e segurança. Nesse exemplo, luz é uma característica opcional do sistema e, portanto, tanto o componente *Luz* quanto o conector *connMicroondasLuz* são opcionais também. Assim, *Luz* e *connMicroondasLuz* são variantes de um ponto de variação na arquitetura da linha de produto. A arquitetura da linha de produto pode seguir algum estilo arquitetural, ajudando a definir como módulos e subsistemas são organizadas para a composição do sistema [SEISPLv5].

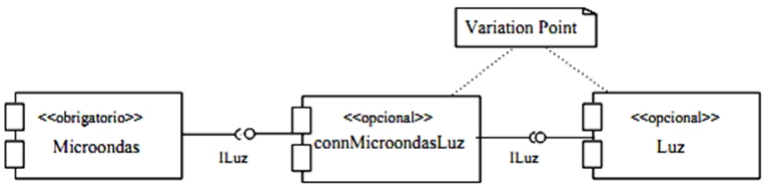



Figura 2.9: Exemplo Arquitetura de Linha Produto baseada em Componentes

## 2.7 Mapeamento de características para elementos arquiteturais

A transformação de características para elementos arquiteturais é uma etapa considerada na implementação na arquitetura LPS. A transformação pode ser realizada de uma característica para um elemento arquitetural [Zhang08] ou um grupo de características para elementos arquiteturais [Sochos06]. Existem vários métodos na literatura que apoiam esta transformação, um deles é o método *Feature-Achitecture Mapping* (FArM) [Sochos06]. Dado um modelo de característica como entrada este método propõe o refinamento visando criar uma arquitetura LPS.

### 2.7.1 Método FArM

O FArM considera a aplicação de quatro etapas progressivas e iterativas. Cada etapa tem como entrada o modelo de característica e a saída um modelo refinado, que sucessivamente é transformado, até a obtenção da arquitetura LPS. As principais motivações do método estão ligadas a requisitos não funcionais, como desempenho, que representam qualidades sistêmicas e que não podem ser representadas por um único componente arquitetural.

### 2.7.2 Método AO-FArM

O método AO-FArM (*Aspect-oriented Feature-Architecture Mapping*) proposto por Tizzei et. al. (2012a) é uma extensão ao método FArM original, onde mapeamento de características base e transversais apoia a criação de arquiteturas de LPSs baseadas em componentes e aspectos por meio de uma série de transformações aplicadas no modelo de características e na visão de características Orientada a Aspectos (OA).

O método AO-FArM utiliza dos conceitos de aspectos utilizado para separar interesses e consequentemente ajuda na transformação de características para elementos arquiteturas utilizando aspectos.

O conceito de **interesse** é definido como sendo foco que um dos *stakeholders* considera importante em um sistema [Robillard07]. Alguns desses interesses são transversais, são propriedades com escopo abrangente e geralmente presentes em diversos módulos em um sistema de software [Kiczales97].

Para a aplicação do AO-FArM é necessário anteriormente a primeira etapa ter o modelo de característica e outros artefatos como a especificação de requisitos e a **Visão de Característica OA**. Este último modelo é decorrente de fases anteriores remetendo a um forte emprego de técnicas do Desenvolvimento Orientado a Aspectos (AOSD) para garantir a identificação e composição de casos de uso ligados a interesses transversais, bem como a modelagem de requisitos e características utilizarem destes conceitos também. Em suma, os conceitos são:

**Interesse transversal** – são propriedades com escopo abrangente e geralmente presentes em diversos módulos. A definição do uso do tipo de relacionamento **entrecorta** é usado para identificar esse tipo de interesse nos modelos.

**Interesse base** – são propriedades que sofre algum tipo de mudança da ligação de um interesse transversal com base, mas que são partes do sistema que não tem aspectos.

A Visão de característica OA depende da execução de um conjunto de atividades aplicadas na fase de análise de requisitos, porém, orientada a aspectos e características. As atividades desta fase são apresentadas na Figura 2.11.

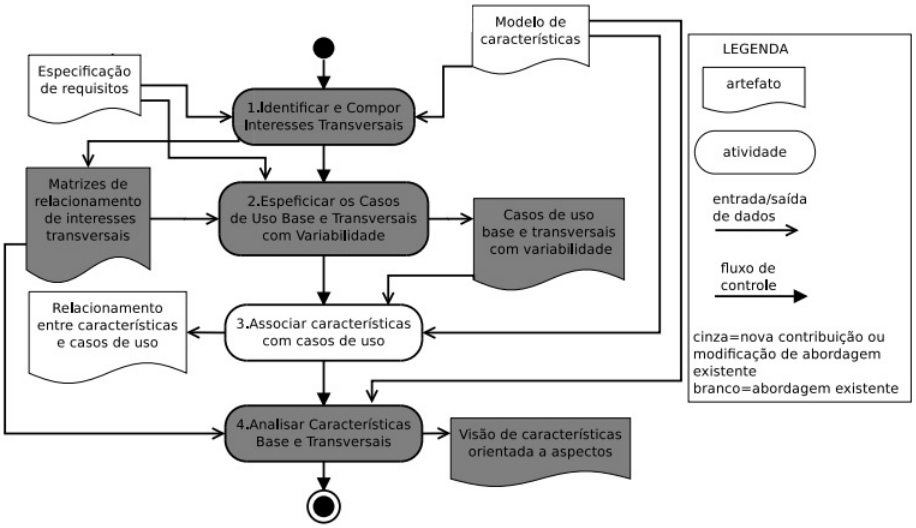



Figura 2.10: Atividades da fase de análise de requisitos orientada a aspectos e características de LPSs (Tizzei et. al. 2012a)

A modificação aplicada por Tizzei (2012a) pode ser notada na Figura 2.10. Algumas atividades modificam a abordagem como nas atividades em cinza. As abordagens modificadas contêm extensões às abordagens existentes propostas na literatura [Tizzei12a].

### Notação no modelo de característica versus Visão de característica OA

A notação utilizada nos diagramas de característica e na Visão de característica OA é representada para identificar relações entre elementos que representam interesses (transversal»base, base »base ou transversal »transversal). Esta relação utiliza uma notação de forma semântica ao modelo de característica original, contudo, a representação do modelo Visão de característica OA não é hierárquica como no modelo de característica original, pois o objetivo principal é oferecer uma visão racional que influencia as características base das características transversais.

Na Arquitetura de Linha de Produto, o modelo de característica transformado tem notações semânticas também, incluindo os estereótipos **«usa»** para indicar dependência

entre componentes, **«entrecorta»** para simbolizar um componente que é influenciado por outro componente que contém aspecto, ou seja, um componente base é entrecortado por outro componente com interesse transversal. Caso um elemento com interesse transversal entrecorte mais de uma única vez outro elemento é utilizada o estereótipo **«precedida-por»** para identificação deste tipo de relação [Tizzei12a].

Estas notações são usadas para dar apoio a uma visão do modelo de característica de forma semântica do modelo original da LPS para um modelo que destaque e diferencie a composição de características que são transversais das que são base.

A Figura 2.11 mostra o uso da notação no diagrama Visão de característica OA. Note que o diagrama apresenta as características transversais que entrecortam características base. Ainda na Figura 2.11, as características bases Microondas e Luz são entrecortadas pela característica de interesse transversal Log.

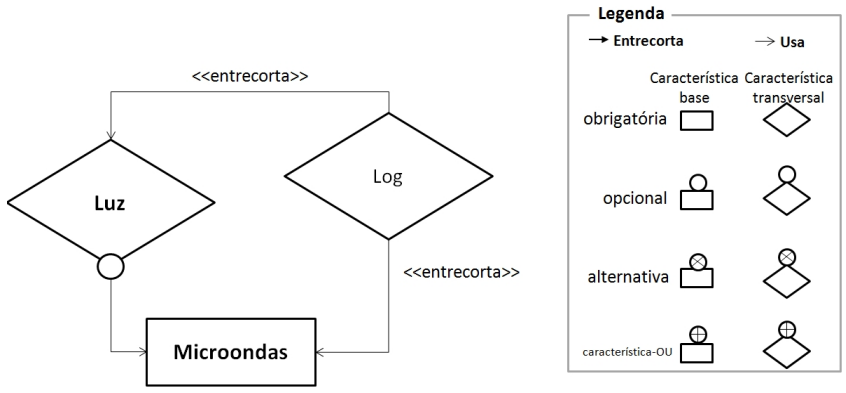



Figura 2.11: Visão de características orientada a aspectos

### Visão de características OA

De acordo Tizzei et. al. (2012) a criação da visão de características OA depende do modelo de características e da identificação dos interesses transversais. A aplicação dos passos descritos por Tizzei et. al. (2012), juntamente com o modelo de característica pode-se iniciar as etapas seguintes relativas ao método.

O modelo de características e a visão de características OA são transformados iterativamente dando origem ao modelo de características transformado e a visão de características OA transformada. São esses modelos transformados que guiam a identificação dos componentes base e transversais, bem como dos relacionamentos entre os componentes, que compõem a arquitetura de LPS.

A Figura 2.12 mostra as transformações, representadas como atividades, e artefatos envolvidos no mapeamento das características base e transversais para a arquitetura de LPS. As duas primeiras atividades desta fase são iguais às descritas no método FArM

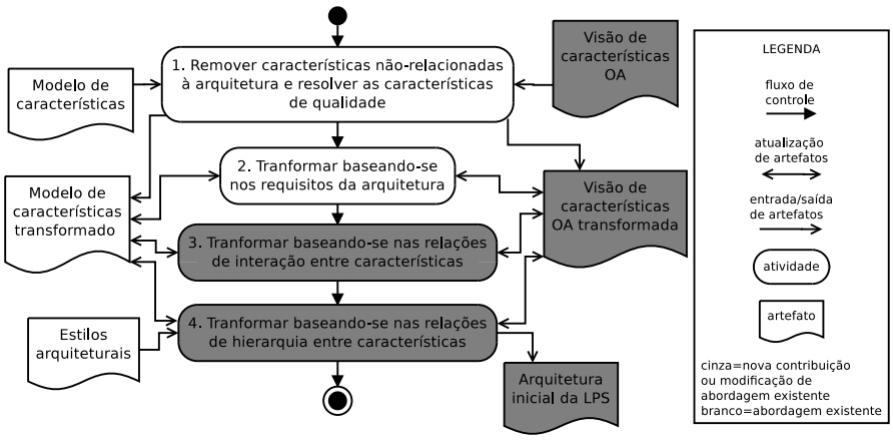



Figura 2.12: Etapas da aplicação da extensão (AO-FArM) das transformações do FArM (Tizzei et. al. 2012b)

e a única modificação é que as transformações aplicadas ao modelo de característica transformado também são aplicadas na visão de características OA transformada. Por exemplo, se uma característica é adicionada ao modelo de característica transformado, ela também deve ser adicionada à visão de características OA.

## 2.8 Modelo COSMOS* e Programação Orientada a Aspectos

Modelos de implementação de variabilidade retratando aspectos transversais e modularização tem sido discutido em diferentes trabalhos [Dias10, Tizzei11]. Tais trabalhos contribuíram no âmbito da modularização do sistema ao usar mecanismos de aspectos para oferecer o desacoplamento entre módulos, e ao apoiar a implementação das variabilidades por meio de componentes e aspectos. Primeiramente, o modelo COSMOS* oferece uma representação no contexto de LPS através de componentes e que posteriormente foi estendido para COSMOS*-VP para minimizar o acoplamento entre componentes a partir de interesses transversais usando aspectos.

### 2.8.1 Modelo COSMOS*

O COSMOS* é modelo de componentes existente [Gayard08], desenvolvido pelo grupo de pesquisa SED (*Software Engineering and Dependability*), do Instituto de Computação da UNICAMP. Esse modelo oferece diretrizes para codificar elementos arquiteturais e separa explicitamente a especificação da implementação dos componentes. A especificação do componente define seus serviços por meio de interfaces providas e suas dependências de outros serviços por meio de interfaces requeridas. A implementação do componente

é encapsulada, restringindo o acesso externo de forma a evitar dependências indesejadas entre componentes [Silva03]. O modelo COSMOS* também define um modelo de conector, cujo objetivo é intermediar a comunicação entre os componentes.

### 2.8.2 Modelo COSMOS*-VP

A extensão do modelo COSMOS* é chamada de COSMOS*-VP (*Component System Model for Software architectures with Variation Points*). No modelo COSMOS*-VP, os elementos arquiteturais podem desempenhar dois papéis: transversal ou base. Logo, tanto componentes como conectores podem ser transversais ou base [Tizzei11]. Componentes transversais usam aspectos para modularizar características transversais, assim como aspectos que também podem ser usados para implementar a variabilidade [Anastasopoules01, Voelter07, Noda08].

Os componentes transversais são responsáveis por modularizar as variantes visando facilitar a evolução da arquitetura. Conectores transversais, chamados de *connectors-VP*, visam modularizar pontos de variação arquiteturais e intermediar a comunicação entre componentes transversais e componentes base. Os *connectors-VP* usam aspectos para modularizar as decisões relacionadas a escolhas de produtos.

A extensão ao modelo COSMOS* original foi mediante uma pequena modificação, onde um pacote público chamado *aspects* é usado para especificar publicamente os aspectos abstratos de um componente.

### 2.8.3 Programação Orientada a Aspectos

Programação Orientada a Aspectos (POA) oferece mecanismos para a modularização de interesses transversais de um sistema de software [69]. Tais interesses são implementados ao espalhar em módulos do sistema. Os interesses ficam isolados de códigos fontes dos módulos existentes do sistema, o que contribui na modularidade do sistema.

Existem muitas linguagens orientadas a aspectos. Como o atual projeto se baseia em Java, optou-se pelo AspectJ, além disso, diversos trabalhos tem utilizados AspectJ na implementação de LPS [Anastasopoulos04, Figueiredo08 e Alves05].

Os aspectos que serão implementados em um projeto podem ser realizados através de interfaces de programação transversais (XPIs) [Griswold06]. As XPIs ajudam os desenvolvedores na fase de implementação, pois abstrai decisões de projeto detalhado. Através da utilização combinada de componentes, aspectos e XPIs, o projeto de um componente pode especificar os locais onde interceptações são permitidas, tornando esses locais públicos por meio de XPIs.

A Figura 2.14 representa a partir de uma notação estendida do diagrama de componentes do UML 2.4 para representar interfaces transversais e a sua materialização

A Figura 2.13 (a) mostra um exemplo de comunicação entre dois componentes por

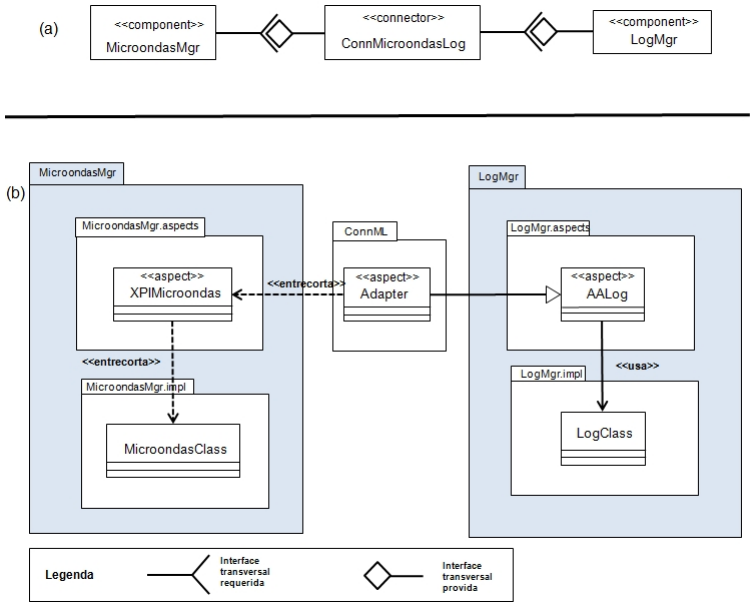



Figura 2.13: (a) Visão arquitetural de uma arquitetura com componentes COSMOS*-VP; (b) Projeto detalhado dos componentes e conector usando o modelo COSMOS*-VP

meio de interfaces transversais. Um componente transversal *LogMgr* entrecorta outro componente base *MicroondasMgr* por intermédio do conector *ConnML*. A interface transversal requerida do componente *MicroondasMgr* visa explicitar os pontos de junção desse componente que podem ser interceptados por outros componentes. Ao expor os pontos que podem ser interceptados, a interface transversal requerida permite que componentes transversais utilizem esses pontos para modificar o comportamento do componente base. Assim, os componentes transversais não quebram o encapsulamento do componente *MicroondasMgr*, já que eles não tem acesso direto às classes de implementação. Neste caso, é mais oportuno antecipar os pontos de junção em uma arquitetura que visará facilitar a evolução da arquitetura a partir dos aspectos.

## 2.9 Resumo do Capítulo

O capítulo teve por objetivo apresentar um conjunto de conceitos relacionados ao foco do nosso projeto sendo Arquitetura Orientada a Serviços, Serviços Web, relacionados a sistemas de monitoração. Os conceitos deste último, foram classificados e resultou em uma taxonomia. Dada a importância desta taxonomia para o projeto, é importante enfatizar que ela nos apoiou durante todo o projeto.

Os conceitos sobre LPS foram também descritos considerando emprego das técnicas para criação da solução proposta a partir de artefatos centrais com foco em família de produtos. O modelo de característica, descrito sob a ótica do diagrama de características

do que é comum e variável em um sistema. De maneira complementar, outros modelos e métodos foram descritos e oferece também apoio a toda a infraestrutura para desenvolvimento dos artefatos (*core assets*) de LPS. A aplicação do Método PLUS e os artefatos gerados, alimentaram um repositório que foi utilizado durante as etapas do projeto.

Após a aplicação do Método PLUS as fases posteriores contempla o mapeamento das características para elementos arquiteturais e um refinamento (já considerada as etapas COSMOS*-VP, FArM e AO-FArM) do modelo de características até se obter um modelo que possa ser transformado para um diagrama de componentes para oferecer uma visão clara na aplicação do desenvolvimento baseado em componentes. Ademais, apoiam no desenvolvimento da solução proposta codificado a partir da FeatureIDE.

O próximo Capítulo abordará o estado da arte do projeto em uma Revisão Sistemática da Literatura.

# Capítulo 3

# Revisão sistemática sobre sistemas de monitoração de serviços Web

Uma revisão sistemática da literatura foi desenvolvida e apoiada nas diretrizes da Engenharia de Software Experimental. Esta revisão é parte de um processo que contempla uma investigação baseada em evidências a partir de questões de pesquisa. Como parte natural deste processo, contribui com o entendimento de conceitos do que é relacionado às questões de pesquisa e que são avaliadas durante a investigação.

As soluções analisadas são classificadas nesta revisão, como estudos primários, de modo que o nosso estudo é classificado como estudo secundário, e foi elaborado por meio do guia original proposto por Kitchenham (2004).

Os métodos aplicados durante a revisão iniciam-se por um conjunto de questões de pesquisa baseadas nos conceitos vistos no Capítulo 2 principalmente sobre sistemas de monitoração de serviços Web. Um protocolo de busca e seleção de estudos com critérios de inclusão/exclusão é descrito. Após, as etapas de avaliação de qualidade sobre os estudos foi realizada. Esta avaliação contribui de forma qualitativa ao classificar e separar os estudos primários considerando a relevância, aplicabilidade e consistência. Os resultados são postos de maneira a identificar a existência de diversos elementos e como eles se interagem entre si dentro de cada estudo sem aferir qualquer julgamento a priori.

Posteriormente, uma discussão é colocada envolvendo todos os estudos primários elegíveis, e considerando a relevância dos fatores identificados. Tais fatores foram quantificados em percentuais e nos permitiu entender o que os estudos primários nos sugerem. O cruzamento destes diversos fatores podem ser vistos através de tabelas, e apoiou nossas conclusões. Ao final, além das conclusões, lacunas não preenchidas são vistas como oportunidades e desafios e possivelmente podem ser solucionadas em estudos secundários.

## 3.1 Questões de pesquisa

As questões de pesquisa que são endereçadas por este estudo são:

- **RQ1.** Quais as principais funcionalidades das abordagens analisadas?
- **RQ2.** Podem ser adicionadas novas funcionalidades de monitoração à solução?

A RQ1 tem como critério de avaliação a taxonomia descrita na Seção 2.2.1 e é representado na Figura 2.4 do Capítulo 2. Onde tem-se o alvo da monitoração, os atributos de QoS, o modo de operação da solução, a frequência de execução e o tipo de notificação. A RQ1 tem por finalidade conhecer e entender mais sobre as principais funcionalidades que podem existir em soluções de monitoração de serviços Web. A RQ1 é dividida nas seguintes sub-questões:

- **RQ1.1.** Quais são os alvos da monitoração?
- **RQ1.2.** Quais atributos de QoS a solução apóia?
- **RQ1.3.** Quais os objetivos da monitoração as abordagens apontam?
- **RQ1.4.** Qual o modo de operação usado?

A RQ1.1 tem em seu critério de avaliação a partir dos alvos eleitos na Seção 2.2.1. São quatro importantes alvos da monitoração, Serviço, Aplicação Servidora, Servidor ou Rede. Para a RQ1.2 o critério de avaliação considerado são os atributos de QoS que fazem parte da recomendação dada pela W3C [W3C03]. Embora, usamos as diversas taxonomias e categorias levantadas para este estudo como fontes de entendimento, tendo principalmente como forte apoio o padrão de ISO/IEC 9126-1.

A RQ1.2 ainda tem em sua classificação como resultado do questionamento sobre a forma como os atributos de QoS são descritos, previstos e atributos de QoS que são descritos, previstos e monitorados efetivamente. Note a diferença, atributos descritos no artigo são aqueles mencionados como importantes na monitoração, mas, sem prévia proposta de solução e meios de monitoração. Atributos previstos são atributos que poderiam entrar na solução e há propostas de monitoração para tais atributos, mas, não necessariamente há evidências que caracterizam uma solução por meio de estudos de caso ou outros critérios de avaliação. Se os atributos de QoS são descritos, são previstos meios de obtê-los e são efetivamente testados com a solução proposta, aumenta a qualidade do estudo primário e consequentemente será relevante para o enfoque deste estudo.

Para a RQ1.3 como objetivo da monitoração, os itens mencionados na Seção 2.2.3 do Capítulo 2, onde são descritos alguns de muitos objetivos da monitoração de serviços Web.

Consumidores de serviços Web tem diferentes requisitos e muitas vezes requisitos conflitantes, o que é comum em soluções aplicadas a SOA onde é da natureza a imprevisibilidade. A pergunta RQ2, tem como critério de avaliação se as diferentes abordagens apoiam a adição de novas funcionalidades. A adição de novos atributos de QoS, pode levar eventualmente a adição de novos modos de operação de coleta de dados, com isso, estabelecemos as seguintes alternativas para a RQ2:

- RQ2.1. As abordagens apoiam novas funcionalidades parcialmente por meio de regras de forma estática.
- RQ2.2. As abordagens apoiam novas funcionalidades em tempo de execução.
- RQ2.3. As abordagens não apoiam novas funcionalidades tanto para a novos requisitos ou atributos de QoS.

**Protocolo usado na revisão sistemática da literatura**

A fim de encontrar estudos primários diretamente relacionados às questões de pesquisa, foram definidos os termos de busca. Estes termos foram desenvolvidos usando os mesmos passos contidos em Kitchenham (2004), e são sumarizados na Tabela 3.1.

Tabela 3.1: Termos de busca

| | |
|---|---|
| População | ( web services OR ws OR<br>services OR web service<br>OR distributed systems) |
| Intervenção | AND (monitoring attributes qos OR<br>attributes qos monitoring OR<br>qos monitoring OR quality of<br>service monitoring OR requirements monitoring<br>OR monitoring requirements OR<br>monitoring constraints OR monitoring OR qos) |
| Resultado | AND (approach OR proposal OR purpose<br>OR management) AND<br>(OR static OR dynamic OR runtime OR online OR adaptation) |
| Contexto | AND (SOA OR services oriented architecture<br>OR SOC OR services oriented computing OR service selection) |

A pesquisa foi realizada obtendo os estudos relacionados ao tema nas seguintes bibliotecas digitais: ACM Portal, IEEE, Scopus e Springer Link. Estas bibliotecas digitais também foram usadas em trabalhos relacionados como [Benbernou10, Yu08, Cabrera12].

Após a execução da pesquisa, dos resultados obtidos foram analisados primeiramente título e *abstracts*. Esta análise contribui com a percepção dos estudos relacionados aos

nossos objetivos para a seleção de estudos relevantes. Após esta primeira etapa, à aplicação sobre os estudos relevantes consistiu dos seguintes passos descritos na Tabela 3.2, bem como a execução do critério de seleção do artigo baseado na Tabela 3.3.

Tabela 3.2: Procedimentos de seleção

| **Passos** | **Descrição** |
|---|---|
| Passo 1.1 | Leitura em largura (títulos e *abstracts*) |
| Passos 1.2 | Seleção de artigos relevantes |
| Passo 1.3 | Verificação da integridade |
| Passo 1.4 | Leitura em profundidade avaliação crítica e da qualidade do artigo |
| Passo 1.5 | Análise de acordo com os critérios de inclusão e exclusão |
| Passo 1.6 | Remoção de artigos duplicados |

Tabela 3.3: Critérios de inclusão e exclusão

| |
|---|
| ...estar disponível entre os recursos selecionados; |
| ...estar escrito em inglês ou português |
| ...ter algum estudo relevante sobre sistemas de monitoração de serviços web |
| ...ter alguma relação entre monitoração de serviços web e atributos de qualidade de serviços |
| ...descrever claramente sua metodologia |
| ... ter dados suficientes para uma análise conclusiva |
| ...ter sido concluídos e completos |
| ...ser uma solução ou uma metodologia de monitoração de qualidade de serviço aplicado em web services |
| ...os artigos devem ter sido publicados em conferência ou revista |
| ...não ser um artigo do tipo *Survey* |

O espaço temporal de pesquisa considerou a obtenção de artigos nos últimos 5 anos, devido ao dinamismo da computação, o avanço tecnológico e o tópico de pesquisa ser exaustivamente explorado na comunidade. Com isso, os artigos devem ter sido publicados entre 2008 a 2012, como critério de inclusão e exclusão.

## 3.2 Avaliação da Qualidade

A fim de avaliar a qualidade dos estudos primários, foi definido baseado em Kitchenhan [Kitchenham04] as seguintes Questões de Qualidade:

- QQ1. Atende a algum padrão estabelecido de monitoração de atributos de qualidade de serviço (e.g. W3C[W3C03], ISO/IEC-9126 [Iso01])?

- QQ2. Existe avaliação da qualidade a partir da verificação e validação coerente sobre a metodologia proposta (e.g. observação, estudos de caso, experimentos controlados, provas de conceito)?

- QQ3. Os requisitos de monitoração utilizados são claramente descritos a partir de uma avaliação para cada atributo de qualidade de serviço estudado?

Para as respostas às questões de avaliação de qualidade definimos a pontuação sendo Sim como resposta, um ponto (1), enquanto que Parcial meio ponto (0.5), e Não a pontuação zero (0).

- **QQ1:** Os artigos analisados podem ter um ou mais atributos contidos nos padrões e taxonomias abordados na Seção 2.2.1. No entanto, artigos que avaliam propriedades que relacionam ao menos um atributo contido no padrão receberiam um ponto (1). Enquanto a não avaliação de qualquer atributo levaria zero (0). Atributo fora do padrão levaria uma pontuação parcial desde que se consiga obter um grau de relevância para o estudo. Neste último caso, se a proposta colaboraria de forma a clarear melhor o que os padrões estabelecem como atributos e propriedades e o que de fato é levado em consideração na realidade. Poderemos perceber que pode existir uma expectativa naqueles atributos que não estão contidos em padrões, mas que de alguma maneira, consumidores e provedores de serviços dependem crucialmente deles.

- **QQ2:** Se atributos são mencionados e discutidos no artigo, é necessário que se faça uma verificação e validação da proposta de monitoração contra os atributos que foram mencionados pela proposta. A validação pode ser por meio de estudos de caso, experimentos controlados, aferição por prova de conceitos ou qualquer outro meio que melhor caracterize em uma validação e demonstre os resultados obtidos.

- **QQ3:** A proposta de monitoração deveria levar em consideração que os mecanismos de atuação do monitor são descritos de forma clara, incluindo principalmente os meios pelos quais os atributos são colhidos, analisados e avaliados. Acreditamos que cada atributo QoS é individual e seu tratamento pede uma avaliação especifica e não genérica. Neste caso, o mecanismo de monitoração deveria prover sustentação ao resultado oferecido para cada atributo de QoS descrito na proposta. Esta questão decorre para as seguintes questões para uma apuração mais detalhada:

  - QQ3.1 Há uma descrição detalhada dos atributos de QoS que são monitorados?
  - QQ3.2 Há uma descrição detalhada dos mecanismos de monitoração?
  - QQ3.3 Existe uma validação para o mecanismo de monitoração para cada atributo QoS em questão?

Em geral, a avaliação de qualidade por meio das questões acima, tem por objetivos macros cobrir o máximo de características com base em validação da pesquisa, tratamento de validação, relevância, aplicabilidade e consistência.

## 3.3 Resultados da pesquisa

A busca obteve 112 estudos primários nos recursos escolhidos. Após o procedimento de seleção e aplicação dos critérios de inclusão e exclusão, foram selecionados 33 estudos primários para uma análise com mais acurácia. A Tabela 3.4 apresenta a distribuição dos estudos separados por recurso pesquisado.

Tabela 3.4: Distribuição dos estudos separados por recurso pesquisado

| Recurso | Resultado da pesquisa | Estudos primários relevantes |
|---|---|---|
| ACM Portal | 14 | 7 |
| IEEE Xplorer | 47 | 18 |
| Springer Link | 34 | 4 |
| Scopus | 17 | 4 |

### 3.3.1 Resultados e discussão da avaliação de qualidade

Aplicação da avaliação de qualidade foi executada para os 33 estudos primários selecionados, descrita na Seção 3.2. A execução da avaliação foi realizada por dois pesquisadores. Sendo um pesquisador pleno e um júnior.

Foi estabelecida uma classificação separando os estudos primários pela nota obtida na avaliação de qualidade. O critério adotado para a distribuição dos estudos nesta classificação foi a separação de uma nota mínima como regular, uma nota média e uma nota máxima. A nota mínima ou regular representa um estudo primário sugerido como proposta, mas que não é colocada em prática uma avaliação mais concisa. Além disso, atributos de QoS dos modelos mencionados na Seção 2.3 podem não ser identificados nos estudos primários classificando-os com nota mínima. A nota média (Bom) é possível evidenciar algum atributo inserido nos modelos, mas a avaliação é realizada de forma parcial. Na nota máxima (Muito bom) há uma avaliação mais consistente, que responde de forma positiva a maioria de nossas questões de qualidade.

Dos 33 estudos primários, 13 tiveram conceito Bom, para 10 artigos, Muito bom ou Regular, conforme Tabela 3.5 que sumariza valores e classificação.

A Tabela 3.6 apresenta o resultado detalhado desta análise e a nota obtida por questão de qualidade, nota total e a classificação geral como resultado da análise por estudo primário.

Tabela 3.5: Total de estudos primários por classificação

| Classificação | Intervalo | Total estudos primários |
|---|---|---|
| Regular | =< 2 | 10 |
| Bom | 2 > e =<3,5 | 13 |
| Muito bom | > 3,5 | 10 |

A Tabela 3.6 apresenta o resultado da avaliação de qualidade onde cada questão tem sua abreviação por QQ e para cada artigo as respostas (S - Sim, N - Não ou P - Parcial). E as demais colunas representam a nota e a classificação respeitando o critério que estabelecemos anteriormente (R - Regular, B - Bom e MB - Muito bom).

É necessário enfatizar que o restante deste capítulo fará referência em tabelas, resultados e discussão usando o índice e não o código de referência usado na dissertação para referenciar o artigo. Isso foi feito principalmente por uma questão de espaço e visualização. Se necessário, pode-se encontrar o estudo primário por completo na Seção de Referência utilizando o código inserido na primeira coluna da Tabela 3.6.

A Figura 3.1, apresenta o gráfico de avaliação por questões de qualidade. Note que há uma divisão e distribuição dos estudos primários pela resposta a Questão de Qualidade. Através do gráfico da Figura 3.1 é possível avaliar quais questões de qualidade a quantidade de estudos pôde mais ter respostas positivas, negativas ou parciais.

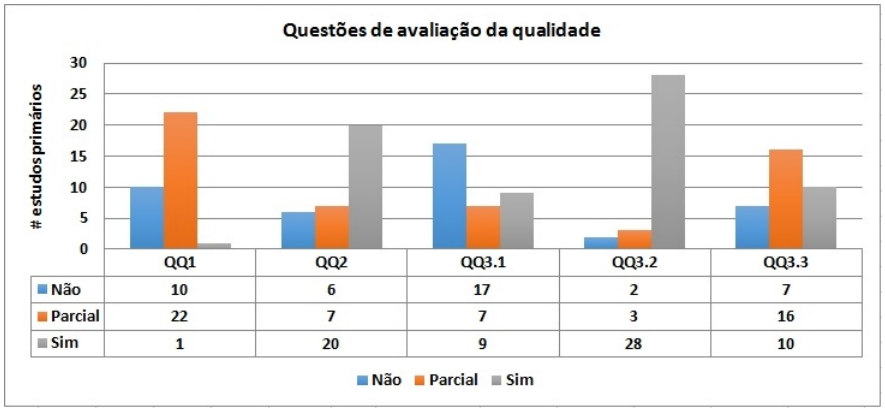



Figura 3.1: Gráfico e tabela da avaliação por questões de qualidade (Avaliação geral)

### 3.3.2 Discussão sobre as questões de qualidade - Avaliação geral

Sobre a QQ1, para 22 estudos não há menção a padrões de qualidade. Entretanto, se faz uso de atributos de QoS contidos em modelos e padrões utilizados como critério de avaliação, mas não necessariamente naqueles recomendados pela W3C nem nos da ISO/IEC-9126-1. Para 10 artigos não há menção a padrão de atributos de QoS, bem como a informação de qual atributo de QoS foi avaliado é omitida, para 1 artigo, o

Tabela 3.6: Análise de avaliação de qualidade para cada estudo primário

| Referência | Índice | QQ1 | QQ2 | QQ3.1 | QQ3.2 | QQ3.3 | Nota | Clas. |
|---|---|---|---|---|---|---|---|---|
| Baresi08 | 5 | N | N | N | S | N | 1 | R |
| Sidibe08 | 13 | N | N | N | S | N | 1 | R |
| Yousefipour10 | 19 | N | N | N | S | N | 1 | R |
| Schoene09 | 7 | P | N | N | P | P | 1,5 | R |
| Baresi08a | 26 | P | N | N | S | N | 1,5 | R |
| Spillner09 | 15 | P | S | N | N | P | 2 | R |
| Tian10 | 18 | N | S | N | S | N | 2 | R |
| Ivanovic10 | 27 | P | P | N | S | N | 2 | R |
| Leitner09 | 28 | N | P | N | S | P | 2 | R |
| Katsaros11 | 31 | P | P | N | S | N | 2 | R |
| Baresi08b | 2 | P | S | N | P | P | 2,5 | B |
| Goel10 | 20 | N | S | N | S | P | 2,5 | B |
| Serhani10 | 23 | N | P | P | S | P | 2,5 | B |
| Zadeh10 | 24 | P | P | P | P | P | 2,5 | B |
| Comes12 | 30 | N | P | P | S | P | 2,5 | B |
| Gmach08 | 1 | N | S | P | S | P | 3 | B |
| Raimondi08 | 3 | P |  | S | N | P | 3 | B |
| Fakhfakh08 | 8 | P | S | N | S | P | 3 | B |
| Guo08 | 9 | N | N | S | S | S | 3 | B |
| Baresi10 | 22 | P | S | N | S | P | 3 | B |
| Romano11 | 25 | P | S | N | S | P | 3 | B |
| Fei08 | 10 | P | S | N | S | S | 3,5 | B |
| Baresi09 | 16 | P | S | N | S | S | 3,5 | B |
| Halima08 | 12 | P | S | S | S | P | 4 | MB |
| Wang09 | 14 | P | S | S | S | P | 4 | MB |
| Godse10 | 21 | P | S | P | S | S | 4 | MB |
| Souza11 | 29 | P | S | P | S | S | 4 | MB |
| Haiteng11 | 32 | P | S | S | S | P | 4 | MB |
| Muller12 | 33 | P | S | P | S | S | 4 | MB |
| Moser08 | 4 | P | S | S | S | S | 4,5 | MB |
| Michlmayr09 | 6 | S | P | S | S | S | 4,5 | MB |
| Artaiam08 | 11 | P | S | S | S | S | 4,5 | MB |
| Wetzstein09 | 17 | P | S | S | S | S | 4,5 | MB |

padrão é percebido ou descrito. Isso decorre do fato da ausência de um modelo com um fim especifico como padrão para uso em ferramentas de monitoração de serviços Web.

Em relação a QQ2, há um grande número de abordagens que efetua algum tipo de validação. Contudo, as que obtiveram nota Parcial, as avaliações não consideram atributos de QoS ou validam de forma superficial demonstrando resultados que não é possível

entender se a proposta é relevante como meio de monitoração. Nas demais que obtiveram Não, nota-se que não há evidências claras de uma validação mais coerente.

Sobre a QQ3, não há uma descrição detalhada, nem modelos de atributos de QoS são seguidos. Entretanto, mecanismos de monitoração são bem descritos. Disso resulta que para a QA3.3, a nota é Parcial para a maioria considerando a dificuldade em entender que mecanismos são descritos, mas não necessariamente ocorre a validação de tais mecanismos contra os atributos de QoS mencionados e previstos no estudo. A partir da ausência de modelos de atributos de QoS identificada na QQ1, os requisitos para a ferramenta de monitoração não irão obedecer um conjunto de atributos de QoS, a validação é feita para um número mínimo de atributos em atendimento ao fim especifico que a ferramenta de monitoração foi proposta.

Em resumo, os artigos com conceito Muito Bom se destacam principalmente por descrever os atributos de QoS usados de forma clara, embora não informe qual padrão de qualidade seja usado, mecanismos de monitoração são bem detalhados e bem validados compondo uma proposta com alto nível de relevância para o presente estudo.

Os artigos em geral, são propostas de monitoração com uma boa relevância para o presente estudo.

### 3.3.3 Fatores identificados na avaliação da qualidade

A avaliação da qualidade para as abordagens detectou que:

1. Os estudos não descrevem sucintamente atributos de QoS o que torna difícil de mapeá-los;

2. Nem sempre foi possível identificar uma validação da proposta por experimentos controlados, estudos de casos ou quaisquer outros meios de validação;

3. Na validação da proposta como um todo, são considerados um ou dois atributos de QoS, mesmo o estudo mencionando atender mais atributos além daqueles que foram validados.

4. Não há na maioria um perfil ou modelo tanto de monitoração quanto de atributos de QoS, o que dificulta o entendimento da validação da proposta em estudos de caso e experimentos controlados;

5. Em nenhuma proposta foi mencionado algum padrão de qualidade como ISO [Iso01] ou requisitos definidos pela W3C [W3C03];

## 3.4 Resultados por questões de pesquisa

### 3.4.1 RQ1. Quais as principais funcionalidades das abordagens analisadas?

Nesta seção, são discutidas as principais funcionalidades utilizadas nas abordagens analisadas.

A RQ1 está ligada a taxonomia definida na Seção 2.2.1 do Capítulo 2 e foi subdividida em subquestões descritas na Seção 3.1 deste Capítulo.

O resultado da análise dos estudos primários referente a cada subquestão é descrito a seguir.

#### *RQ1.1 Quais são os alvos da monitoração?*

O alvo da monitoração determina onde o mecanismo do Monitor atua. Com isso, é necessário entender o que de fato é monitorado e em qual nível ocorre para cada trabalho analisado a partir de uma classificação. Isso pode dizer mais sobre alvo da monitoração, principalmente àquelas abordagens que interceptam dados de mensagens de uma comunicação entre provedor e consumidor. A Tabela 3.7 apresenta esta classificação sobre o que é monitorado e em qual nível isso ocorre.

Em alguns casos, autores representam estes mesmos níveis de forma diferente. Como nível de comunicação, nível de execução e nível de composição [12]. Ora se a monitoração ocorre em qualquer destes níveis, então ocorre exatamente sobre o serviço ou sobre as mensagens inerentes a ele, o que se conclui, portanto, que ambos os níveis fazem parte do nível de serviço.

O modelo utilizado neste estudo para identificar os pontos nas propostas é semelhante ao aplicado em [14] onde a monitoração ocorre nas mensagens de requisição/resposta, na aplicação (estado e evento) e sobre os recursos (estado). As abordagens que interceptam mensagens no nível de comunicação podem atuar tanto no lado do provedor quanto do consumidor. O foco são as mensagens que são interceptadas, pode-se concluir neste caso, que o ponto de monitoração atua em nível de serviço. As mensagens estão já formatadas no padrão XML e estão em alto nível de comunicação. Conclui-se que quando a monitoração ocorre interceptando em baixo nível, ou analisando pacotes Ping ou protocolo SNMP é em um nível de rede. Com isso, pode atuar em quaisquer dos dois lados da comunicação, seja no provedor quanto no consumidor.

#### *RQ1.2 - Quais atributos de qualidade de serviço são monitorados?*

A RQ1.2 define de forma clara e objetiva, quais atributos a proposta aborda. No caso do serviço Web a W3C [W3C03] recomenda um conjunto de atributos de QoS. Este conjunto de atributos recomendado pela W3C foi utilizado neste estudo.

Tabela 3.7: Classificação do que é monitorado e em qual nível ocorre

| | | | Muito bom | Bom | Regular | Total |
|---|---|---|---|---|---|---|
| O que é monitorado? | Processos BPEL | | [4, 12, 17, 32] | [2, 16, 22] | [5, 26, 27] | **10** |
| | Serviços SOAP | | [11, 14, 29, 30, 33] | [3, 10, 20, 23, 24] | [7, 15, 18, 19, 25, 28] | **16** |
| | Serviços e recursos em geral* | | [6, 11, 14] | [1, 8, 9] | [7, 13, 25, 31] | **10** |
| | Contratos | | [6] | [3, 8, 9, 10, 20] | [15, 18, 25, 28] | **10** |
| | Valores no UDDI | | [21] | | [19] | **2** |
| Em qual nível ocorre? | Sistema | Aplicação servidora | [14] | | [5, 31] | **3** |
| | | Servidor | [14] | [1, 8] | [5, 31] | **5** |
| | Serviço | Consumidor** | [6, 12, 14, 17, 21, 29, 32, 33] | [2, 3, 4, 9, 10, 16, 20, 22, 23, 24, 30] | [5, 18, 19, 25, 26, 28, 31] | **26** |
| | | Provedor** | [6, 12, 14, 17, 21, 29, 32, 33] | [2, 3, 4, 9, 10, 16, 20, 22, 23, 24] | [5, 26, 28, 31] | **22** |
| | Rede | Rede | [6, 11, 14] | [9] | [5, 7, 13, 31] | **8** |

*** Pode incluir a monitoração de log, em baixo nível com ping e SNMP**
**** A monitoração ocorre com a interceptação de mensagens durante a comunicação entre consumidor e provedor ou durante uma invocação do serviço**

Apenas em três abordagens [5, 16, 26] os atributos não são claros o suficiente para mapeá-los. Embora, nas demais os atributos fossem identificados a partir do pressuposto dos elementos e aspectos, formas de atuação, objetivos da monitoração e durante a validação da proposta. A Tabela 3.8 representa os atributos de QoS separados em atributos elencados pela W3C, e outros atributos mencionados pelas abordagens.

Há uma predominância da monitoração de atributos de *performance e availability* como pode ser analisado pela Tabela 3.8. Até mesmo para abordagens que permitem incluir/remover atributos de QoS, tais atributos também são utilizados para completar a validação da proposta.

Tabela 3.8: Atributos de QoS das abordagens

| | Atributos de QoS | Muito Bom | Bom | Regular | * | + | Dif. */+ | Total |
|---|---|---|---|---|---|---|---|---|
| Atributos de QoS (W3C) | *Performance* | [*4, *6, *11, *12, +14, *17, *21, *29, +32, *33] | [*1, *2, *3, *8, +9, *10, 20, 22, 23, +24, *25, *30] | [*7, *13,*15, *18,19, 27, +28, +31] | **19** | **6** | **5** | **30** |
| | *Reliability* | [6, *14] | [2, *3, 30] | | **2** | **0** | **3** | **5** |
| | *Scalability* | [6] | [*1] | | **1** | **0** | **1** | **2** |
| | *Capacity* | | [*1] | | **1** | **0** | **0** | **1** |
| | *Robustness* | [6] | [2] | | **0** | **0** | **2** | **2** |
| | *Accuracy* | [*4, 6, +14, +32] | | [*28] | **2** | **2** | **1** | **5** |
| | *Integrity* | | [2] | | **0** | **0** | **1** | **1** |
| | *Accessibility* | [*11, *21, 29] | | [31] | **2** | **0** | **2** | **4** |
| | *Interoperability* | | | | **0** | **0** | **0** | **0** |
| | *Availability* | [*4, *6, *11, +12, +14, *17, *21, 29, +32, +33] | [*1,+9, +10, 20, 22, 23, 30] | [*7, *15, *18, 19, +28, 31] | **9** | **7** | **7** | **20** |
| | *Security* | [*11, *14] | | | **2** | **0** | **0** | **2** |
| Outros atributos de QoS | *Dependability* | [4, 6, *11] | | | **1** | **0** | **1** | **2** |
| | *Regulatory* | [*11] | | | **1** | **0** | **0** | **1** |
| | *Reputation* | [*21] | [+9] | [+19] | **1** | **2** | **0** | **2** |
| | *Satisfation* | | [+9] | [+19] | **0** | **2** | **0** | **2** |
| | *Processing time* | | [+23] | | **0** | **1** | **0** | **1** |
| | *Execution time* | | [*24] | [27] | **1** | **0** | **1** | **2** |
| | *Cost&payment* | [6] | [30] | | **0** | **0** | **2** | |
| | *Safety* | | [22] | | **0** | **0** | **1** | **1** |
| | Outros aspectos | | [16, 20] | [5, 26, 31] | **0** | **0** | **4** | **4** |

**(*) Atributo mencionado como possível de atingir e descrito formas de operar, validado/verificado**
**(+) Atributo mencionado no artigo e descrito parcialmente, mas não validado e verificado**
**( ) Atributo somente mencionado no artigo mas não é possível prevê-lo no estudos primários**
**Safety → indica se o estado do sistema alvo é seguro ou se mantém seguro em um dado evento ocorrido [22]**
**Security → indica se existem propriedades de segurança (e.g. transmissão das mensagens entre provedor e consumidor) [11, 14]**

### *RQ1.3. Quais os objetivos da monitoração as abordagens apontam?*

Os objetivos da monitoração podem ser identificados a partir de funcionalidades básicas que são fundamentais na monitoração. Para a maioria das propostas analisadas, há algum tipo de notificação, seja em formato de log ou disparo de alguma ação que emite algum tipo de alerta [6, 8, 11]. Outras abordagens tem funcionalidades para determinar o comportamento do Monitor para atender a objetivos específicos.

Em resumo, foram encontrados três grandes objetivos: Notificação, aprimorar a seleção

Tabela 3.9: Objetivos do monitor

| Objetivos da monitoração | Muito bom | Bom | Regular | Total |
|---|---|---|---|---|
| Notificação | [6, 11, 12, 14, 17, 21, 29, 33] | [3, 8, 9, 16, 23] | [25, 13, 19, 28, 31] | 18 |
| *Broker* | [21, 32] | [20, 30] | [7, 15, 19, 28] | 8 |
| Adaptação dinâmica | [4, 29] | [1, 2, 10, 20] | [5, 18, 26] | 10 |

do *Broker* ou somente melhorar a seleção de serviços equivalentes e adaptação dinâmica ou recuperação dinâmica em situações onde existem falhas no serviço monitorado. A Tabela 3.9 apresenta os objetivos do monitor. Os objetivos mencionados na Tabela 3.9 foram vistos na Seção 2.2.3 do Capítulo 2.

### *RQ1.4. Qual o modo de operação usado?*

O modo de operação determina como será realizada a coleta de dados do serviço. A Tabela 3.10, apresenta o que cada abordagem apoia como modo de operação.

Tabela 3.10: Modo de operação as abordagens apoiam

| Modo de operação | Muito bom | Bom | Regular | Total |
|---|---|---|---|---|
| Invocação | [4, 6, 17, 21, 32] | [2, 20] | | 7 |
| Instrumentação | [4, 14, 32] | [2, 10, 16, 20, 23] | | 8 |
| Interceptação | [4, 12, 14, 17, 29, 32, 33] | [2, 3, 10, 13, 16, 20, 23, 30] | [19, 26, 28] | 18 |
| Inspeção | [6, 11, 14, 33] | [8, 9, 20, 23, 24] | [7, 13, 18, 25] | 13 |
| Desconhecido* | | [1] | [**5, 15, 27, **31] | 5 |
| (*) Nem todos os estudos primários foi possível apurar o modo de operação utilizado no Monitor, em outros casos pode ser considerado combinações entre os modos de operação.<br>(**) Baseada em plugins | | | | |

Na maioria das propostas analisadas, há uma predominância no uso da interceptação baseado em técnicas de *Programação Orientada a Aspectos* (AOP[1]) [2, 4, 10, 14, 16, 22, 26]. Algumas não mencionam qual o modo de operação usado [1, 15, 27]. Por outro lado, algumas são tão flexíveis como as que usam linguagens e plugins para atingir este

[1]AOP - *Aspect-Oriented programming*

objetivo. Nestes casos não é possível identificar o modo de operação, pois cada plugin tem sua autonomia de funcionamento [5, 31].

### 3.4.2 RQ2 - Podem ser adicionadas novas funcionalidades de monitoração à abordagem?

A RQ2 permite identificar nos estudos primário um nível de flexibilidade do monitor operar. Ao tratar de diferentes atributos de QoS, se espera que mecanismos de monitoração tenha flexibilidade nas suas formas e características, mesmo que isso seja estático, ao passo que quando ocorre em tempo de execução, tais abordagens tem um perfil versátil, assumindo dinamicamente comportamentos específicos de acordo com o ambiente em questão.

Tabela 3.11: Nível de flexibilidade a requisitos do monitor

| Nível de flexibilidade | Muito bom | Bom | Regular | Tot. |
|---|---|---|---|---|
| As abordagens suportam novas funcionalidades parcialmente por meio de regras de forma estática | [17, 33] | [2, 8, 10, 16, 20, 22, 24, 25, 30] | [5, 15, 26, 28, 31] | 16 |
| As abordagens suportam novas funcionalidades em tempo de execução | [14, 29] | | | 2 |
| Não suportam extensão a novos requisitos ou atributos de QoS | [4, 6, 11, 12, 21, 32] | [1, 3, 9, 23,] | [7, 13, 18, 19, 27] | 15 |

Por meio da Tabela 3.11, é possível analisar o comportamento dinâmico de abordagens como em [14, 29]. Em [14] os autores exploram diferentes modos de implementação (e.g. instrumentar ou interceptar), para criar *probes* específicos ao contexto para atingir diferentes atributos de QoS. A extensão chega a níveis complexos ao misturar com linguagens como WSCDL que permite definir novas regras de monitoração.

Em [29] um serviço pode ser implantado em tempo de execução no módulo responsável contendo diferentes características de monitoração. Embora, enfatize que tais serviços façam uso de informações colhidas a partir da interceptação (*QoSInterceptors*). Algumas abordagens, a extensão a novas funcionalidades é por meio de linguagens [2, 5, 8, 10, 15, 16, 17, 20, 22, 30, 33] e permite a adição de novas regras. Mas, o monitor em si, opera baseado em um perfil estático sem possibilidade de acrescentar novas características ao monitor. Com exceção de abordagens que utilizam de plugins onde é possível adicionar novas funcionalidades. Cada plugin pode ser visto como uma extensão à ferramenta de monitoração, como [5, 8, 16, 31].

Em [5, 16] os autores também mesclam da flexibilidade de novas regras e de novas formas de inserir novas funcionalidades no monitor. Porém, isso não ocorre em tempo de execução. Em [31], a adição de um plugin que possa monitorar qualquer atributo e de qualquer forma, é possível quando se faz uso de ferramentas como *Nagios*. Mas, se há um novo plugin, é necessário que a adaptação ao núcleo central seja de forma estática. *Nagios* é ferramenta de monitoração usada por administradores de rede.

## 3.5 Discussão por questão de pesquisa

**RQ1. Quais as principais funcionalidades das abordagens analisadas?**

***Sobre o alvo da monitoração:***

Para 76% das abordagens atua sobre mensagens recebidas pelo consumidor. Contudo, para este conjunto de abordagens, não é possível determinar um valor de um dado atributo de QoS somente analisando serviços ou mensagens. Ou seja, na prática é necessário monitorar o serviço como alvo principal e também nos níveis por onde qualquer desvio que gere degradação no atributo de QoS. Assim, a monitoração deveria ocorrer nos três níveis, rede, serviço e sistema permitindo obter um valor do atributo de QoS com muito mais precisão. Foram identificadas 8 propostas (24%) que efetuam uma monitoração considerando contexto geral além de aplicá-la sobre o serviço.

***Sobre o modo de operação do monitor:***

Para 18 das 33 propostas (54% dos estudos), o modo de operação usado é a Interceptação em modo de execução não intrusivo. Isso decorre de que a maioria das propostas efetua uma monitoração pontual e bem específica atuando diretamente no nível de comunicação. Utiliza-se de técnicas para interceptar mensagens de *request/response*, podendo ser na maioria dos casos, partindo do consumidor com o objetivo de detectar violações de acordos SLAs, gerando notificações e alarmes.

Um fator limitante das abordagens que usam técnicas AOP para interceptação de dados é que tais abordagens dependem de uma dada plataforma para operar. Assim, estas abordagens utilizam de um arcabouço de recursos (e.g. aplicação servidora, dependente de sistema operacional especifico, ActiveBPEL), ao invés de utilizar de uma simples máquina virtual Java (*Java Runtime Virtual Machine*).

A **independência de plataforma** é um fator importante, pois determina a capacidade da proposta de monitoração em operar de forma independente de aplicações servidoras.

***Sobre os objetivos do monitor:***

Para 54% das propostas analisadas o objetivo principal é notificar provedores ou consumidores alertando-os sobre algum evento relacionado a atributo de QoS. Estes objetivos da ferramenta de monitoração podem ser combinados, para atender a diferentes expectativas. Como exemplo, consumidor ser notificado que um dado serviço parou de funcionar e oferecer um serviço equivalente com base no aprimoramento da seleção de serviços do UDDI.

**Em resumo, sobre a RQ1.1, R1.2 e R1.3 (exceto atributos de QoS) evidenciamos que:**

1. A monitoração ocorre sobre processos BPEL;

2. O protocolo SOAP é explorado por todas as abordagens analisadas, mesmo que há uma demanda crescente por outros protocolos e serviços como RESTful;

3. Novas tendências que demandam novas formas de monitoração como Cloud Computing exigem um dinamismo, porém apenas a abordagem [25] retrata tal contexto;

4. A inexistência de um modelo de atributos de QoS e perfil sobre a monitoração dificulta o entendimento da validação de cada proposta;

5. Modos de operação do Monitor se baseiam principalmente em interceptação de mensagens, efetuando notificações quando algum desvio é detectado.

6. Há uma forte dependência de plataforma para a monitoração ser executada

***Sobre os atributos de QoS - RQ1.4. Quais atributos de QoS são monitorados?***

As propostas [2, 16, 8, 31, 5, 25, 14, 29, 22] são flexíveis a qualquer atributo que se deseja monitorar mediante as adaptações necessárias. Como exemplo, utilizar de novos plugins, re-escrever novos serviços ou inserir novas restrições por meio de linguagens. Contudo, estas mesmas abordagens validam a proposta com base em apenas um atributo ou nenhum, como pode ser visto na Tabela 3.12. Isto ainda inclui as que permitem a troca de atributos de QoS em tempo de execução. Para as abordagens [16, 31, 5, 22] como prever novos atributos sem que isso tenha sido de fato comprovado na avaliação da proposta? Há dúvidas neste caso, de como pode ser previsto novos atributos de QoS, tendo por princípio que cada atributo tem sua individualidade e sua forma de atuação, a flexibilidade de fato não é trivial.

Tabela 3.12: Abordagem dinâmica por atributos de QoS, verificação, validação e aspectos

| Estudo Primário | Abordagem dinâmica | Atributo mencionado | Atributo validado e verificado | Conceito |
|---|---|---|---|---|
| **16** | Qualquer atributo previsto | Aspectos funcionais e não funcionais | Nenhum | B |
| **8** | Qualquer atributo previsto | *Performance* | *Performance* | B |
| **31** | Qualquer atributo previsto | *Performance, Accessibility, Availability,* outros aspectos | Nenhum | R |
| **5** | Qualquer atributo previsto | Aspectos funcionais e não funcionais | Nenhum | B |
| **25** | Qualquer atributo previsto | *Performance* | *Performance* | B |
| **14** | Qualquer atributo em tempo de execução | *Performance, Accuracy, Availability, Security* | *Security* | MB |
| **29** | Qualquer atributo em tempo de execução | *Performance, Accessibility, Availability* | *Performance* | MB |
| **2** | Qualquer atributo em tempo de execução | *Performance, Reliability, Robustness, Integrity* | *Performance* | B |
| **22** | Qualquer atributo em tempo de execução | *Performance, Availability, Safety* | Nenhum | B |

**Em resumo, sobre a RQ1.4 é possível evidenciar que:**

1. Há uma predominância de monitoração sobre atributos de QoS *performance e availability*;

2. As abordagens validam suas propostas mediante uso de atributos de QoS observáveis cujo aspectos sejam eventos temporais (sigla E.T.), conforme Tabela 3.14;

3. Mesmo para as abordagens dinâmicas e flexíveis, um ou nenhum atributo é considerado na validação o que torna imprevisto lidar com novos atributos que de fato não foram comprovados. A Tabela 3.12 apresenta os estudos que utilizam de uma abordagem dinâmica com os atributos;

4. Por não existir um padrão de fato ou uma entidade/instituição responsável por definir regras de monitoração sobre serviços, as abordagens tem caminhos e estratégias e modelos divergentes entre si. Padrão W3C recomendado, não foi mencionado por nenhuma abordagem. Modelos e taxonomias de trabalhos anteriores auxiliam autores. Haja vista, a grande quantidade de atributos de QoS fora da recomendação dada pela [W3C03].

Tabela 3.13: Atributos de QoS mais monitorados

| Atributo QoS 1 | Atributo QoS 2 | Tipo de coleta | Estudo primário | Total |
|---|---|---|---|---|
| ***Performance*** | *Response time* | E.T. | [1, 2, 4, 6, 8, 10, 11, 12, 15, 17, 20, 21, 29, 32, 33] | 15 |
| | *Latency* | E.T. | [3, 6, 7, 11] | 4 |
| | *Round trip time* | E.T. | [6] | 1 |
| | *Throughtput* | E.T. | [3, 6, 11, 21, 29] | 5 |
| | *Execution time* | E.T. | [6, 11, 15, 24, 27] | 5 |
| ***Availability*** | *Downtime/uptime* | E.T. | [4, 11, 32] | 3 |
| | *Invocation time* | E.T. | [ 22] | 1 |
| | *Delivery time* | E.T. | [17] | 1 |
| | *Timeout* | E.T. | [20] | 1 |
| ***Accuracy*** | *Failed requests/total requests* | E.T. | [4, 28, 32] | 3 |
| ***Accessibility*** | *Total requests sucessfully delivered/interval time* | E.T. | [6, 21] | 2 |
| ***Reliability*** | *Total requests sucessfully delivered valid responses/total number requests during interval time* | E.T. | [11] | 1 |
| ***Security*** | *Scan national vulnerability database* | Estado | [11] | 1 |
| | *Assinatura de serviço* | Estado | [14] | 1 |
| ***Regulatory*** | *availability, accessibility, and reliability* | Estado | [11] | 1 |

***RQ2. Podem ser adicionadas novas funcionalidades de monitoração à abordagem?***

Para 39% dos estudos primários que consideram a flexibilidade de inserir novas regras à esta política, é realizado por meio de linguagens [2, 5, 8,10,17, 14, 15, 16, 20, 22, 27, 30, 33].No entanto,nem todas as propostas analisadas com linguagens, as regras são inseridas de forma dinâmica. Além disso, não é permitido definir um novo modo de operação do Monitor usando uma linguagem nestas abordagens. Para a abordagem [5] que usa plugins juntamente com o uso de linguagens, tanto o modo de operação quanto o comportamento da ferramenta pode sofrer alteração. Entretanto, para esta abordagem [5] a validação não foi o suficiente para prever novos atributos de QoS.

**Em resumo, sobre a RQ2 é possível evidenciar que:**

1. Linguagens que permitem definir regras, semânticas, lógicas e expressões são usadas para dar suporte à flexibilidade;
2. Porém, as linguagens se limitam às regras, mas não a forma do monitor atuar;
3. Plugins e serviços também apóiam a flexibilidade, mas a adaptação de um novo plugin ao núcleo central do monitor não é realizada em tempo de execução.

**Resumo dos tipos de abordagens aplicadas à monitoração mais explorados nos estudos primários**

Considera-se em cada tipo as diferentes formas que uma ferramenta de monitoração pode atuar. O Tipo A é estático e o mais explorado pelos estudos primários. O Tipo B é o que pode existir certa flexibilidade. Para o Tipo A, é possível resumir a análise a partir das questões de modo geral, através da Tabela 3.14, que representa um Perfil de monitoração mais explorado pela maioria das abordagens. A Figura 3.2 representa o Tipo A da Tabela 3.14. Este tipo tem por finalidade a monitoração específica para notificação, ocorrendo em nível de comunicação interceptando as mensagens de forma não intrusiva.

Tabela 3.14: Perfil de monitoração mais explorado

| Característica | Para maioria | No. estudos | % total |
|---|---|---|---|
| Objetivo do monitor | Notificação | 18 | 54 |
| Modo de operação | Interceptação | 18 | 54 |
| Modo de execução | Não intrusivo | 17 | 51 |
| Alvos da monitoração | Consumidor/Serviço | 26 | 78 |
| O que é monitorado? | Serviços SOAP | 16 | 48 |
| Nível flexibilidade | Estática | 18 | 54 |
| Atributos QoS monitorados | *Performance e Availability* | 30, 20 | 90 e 60 |
| Baseado em | Response time e (E.T.) | 15 | 45 |
| Flexibilidade modo operação | Parcial e nenhuma | 16, 15 | 48, 45 |

O Tipo B representado na Figura 3.3 tem por finalidade a monitoração com objetivos semelhantes ao do Tipo A, embora neste caso, faça uso de técnicas para ampliar a flexibilidade do Monitor, agregando a este, novas funcionalidades por meio de linguagens por expressões lógicas e adição de plugins ou serviços. Contudo, esta adaptação por plugins e linguagens não ocorre em tempo de execução.

## 3.6 Desafios e oportunidades

Traçamos a partir dos resultados obtidos e na discussão da revisão sistemática um conjunto de diretrizes para possibilitar a construção de um *framework* com capacidade de agregar

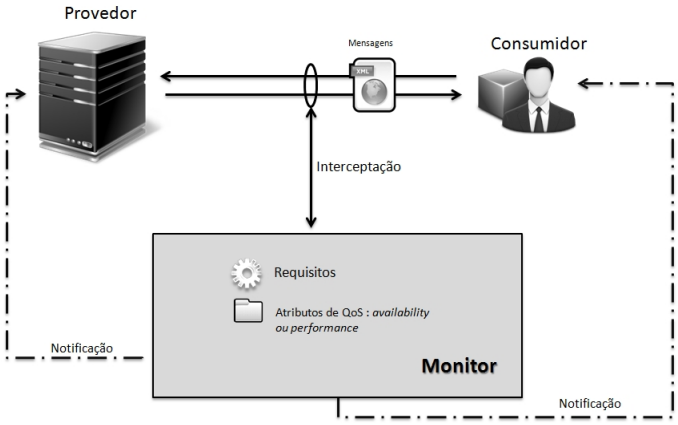



Figura 3.2: Tipo A – Monitoração estática de serviços Web

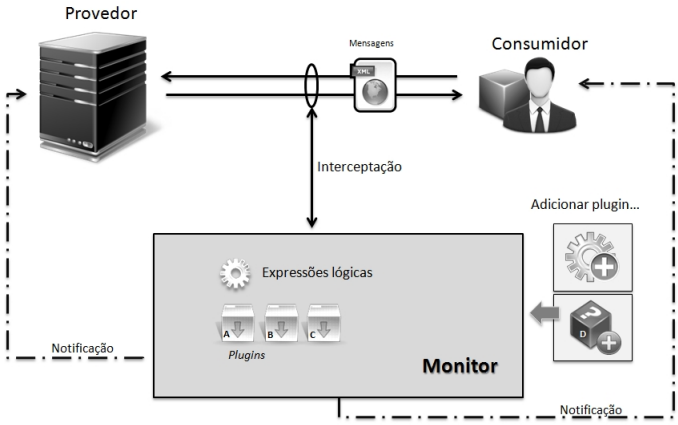



Figura 3.3: Tipo B – Monitoração por meio de plugins, serviços ou linguagens

as lacunas deixadas pelos estudos primários:

1. Definir perfis de monitoração;
2. Tais perfis deveriam agregar flexibilidade às diferentes formas de monitoração, incluindo novos requisitos e novos atributos de QoS, em tempo de execução, leve e transparente;
3. Criar uma solução de monitoração independente de plataforma e portável, com capacidade de ser aplicada na monitoração de recursos de TI e atributos de QoS;
4. Criar uma solução que permita monitorar mais de um alvo simultaneamente;
5. Com a possibilidade de criar perfis, seria desejável que um dado perfil de um monitor poderia atuar em qualquer tipo de protocolo, incluindo serviços SOAP e RESTful. Isso permitira atender as novas tendências que emergem rapidamente, como *Cloud Computing*, serviços embarcados em TVs e *Smartphones*;
6. Criar uma separação clara entre atributos de QoS que se deseja atingir e quais mecanismos serão utilizados;
7. Uma taxonomia clara de atributos de QoS deveria ser levado em consideração nesta separação, incluindo os cálculos e meios necessários de monitorar um atributo, demonstrando ainda, vantagens e desvantagens e as combinações de atributos de QoS conflitantes;
8. Uma taxonomia clara das possíveis formas e características de monitoração deveria ser levada em consideração. Demonstrando o grau invasivo e impacto sobre o serviço que pode acarretar mediante tais combinações;
9. Permitir a criação de níveis de maturidade semelhante ao CMMI que estabeleça uma classificação das abordagens que virão no futuro, definindo um primeiro grau de maturidade para as abordagens estáticas com um conceito regular na avaliação de qualidade até aquelas que contêm um ótimo conceito e abordem dinamicamente com estratégias bem definidas. É perceptível que vários autores, já obtiveram uma evolução no seu grau de maturidade. Por exemplo, Baresi que contém uma gama de trabalhos sobre Monitoração de processos BPEL.

## 3.7 Conclusões

Com o advento de SOA e *Cloud Computting*, serviços tem desempenhado um importante papel ao permitir promover tais arquiteturas e estratégias. Neste cenário, mudanças repentinas são constantes, compreendendo um cenário altamente dinâmico e imprevisível,

principalmente quando se trata de composição de serviços. A concepção de tais abordagens utiliza de padrões de projeto que permitem uma melhor reutilização de código, ao passo que para manter tais abordagens em um contexto baseado em infraestruturas complexas é um desafio, principalmente quando a qualidade do serviço prestado é oferecida como garantia. Entra em cena, a monitoração de serviços avaliando principalmente aspectos não funcionais cujo objetivo são obter métricas de valores de atributos de QoS.

Pelo presente estudo, identificamos que monitoração de serviços Web é *hot topic* na pesquisa o que tem demonstrado a evolução e a tendência e uma grande preocupação em encontrar a solução mais oportuna de monitoração de serviços web. Foram encontrados mais de 100 estudos relacionados a monitoração de serviços Web. Por meio desse trabalho, percebemos ainda, que muito tem sido realizado, e que há um caminho longo a ser percorrido. Com isso, evidenciamos por meio de diretrizes da Engenharia de Software Experimental usando uma revisão sistemática da literatura, que muitos trabalhos abordam serviços com um Monitor específico sobre atributos de QoS, avaliando desempenho e disponibilidade do lado do consumidor.

Este estudo secundário, avaliou 33 estudos primários e na maioria destes não compreendem o atual cenário, não permitindo acrescentar novos atributos de QoS, novas formas e novas funcionalidades ao Monitor. Embora esta flexibilidade fosse importante para nós, tais abordagens contribuem de forma relevante para alavancar e encontrar formas de monitorar atributos de QoS de diferentes maneiras. Na maioria, as abordagens buscam atender a uma preocupação fiscalizadora quando o Monitor é endereçado particularmente como notificação na detecção de violações de SLA, ou seja, como formas de manter as garantias de qualidade, em detrimento a uma abordagem mais colaborativa, quando provedor e consumidor pudessem avaliar seus atributos de QoS, cruzá-los e compará-los sob a perspectiva e interesse de cada lado da comunicação. Visando principalmente na melhoria contínua em um contexto geral. O que de fato contribui com a evolução dessas estratégias como SOA e *Cloud Computting*. Que são essencialmente fundamentais no dia-a-dia no uso corriqueiro de serviços em dispositivos móveis, televisores, serviços *cloud* oferecidos independemente de local, e de tempo ao enviar e receber arquivos das nuvens.

# Capítulo 4

# FlexMonitorWS - Solução proposta

Este capítulo apresenta a solução proposta chamada FlexMonitorWS baseada na criação de uma família de produtos utilizando Linhas de Produtos de Software (LPS) para monitorar diferentes recursos de infraestrutura de TI como alvos, diferentes atributos de QoS de diferentes formas de monitoração.

Primeiramente, foi necessário modelar a variabilidade de software usando um modelo de características focado em sistemas de monitoração de Serviços Web. A conclusão desta primeira etapa, contribuiu com um modelo de característica preliminar ligado a taxonomia descrita na Seção 2.2.1. As etapas seguintes descrevem a aplicação dos métodos ligados a LPS.

## 4.1 Método Pró-ativo proposto para adoção de LPS

Segundo Krueger (2002a), o método pró-ativo é o desenvolvimento de linhas de produto considerando todos os produtos a serem gerados previamente. Neste caso, um conjunto completo de artefatos é desenvolvido do zero para a LPS baseada nas semelhanças obtidas a partir da análise de diversas soluções de monitoração de serviços Web existentes. Para o projeto em questão, adotou-se as fases baseadas no método PLUS proposto por Gomaa (2004), discutido na Seção 2.5.1.

A Figura 4.1 apresenta as fases a partir de um fluxo evolucionário. Note que em cada fase um conjunto de artefatos de entrada foi usado e em consequência a execução da fase gerava-se uma saída de artefatos e assim sucessivamente. Durante as fases, foram utilizados alguns métodos e modelos representados na Figura 4.1 por Ferramentas de apoio.

### 4.1.1 Fase 1 – Modelagem de requisitos

Na Figura 4.1 tem-se a Fase 1 sendo a modelagem de requisitos, onde ocorre uma Análise de Domínio do sistema para se obter uma visão geral sobre a solução a ser desenvolvida.

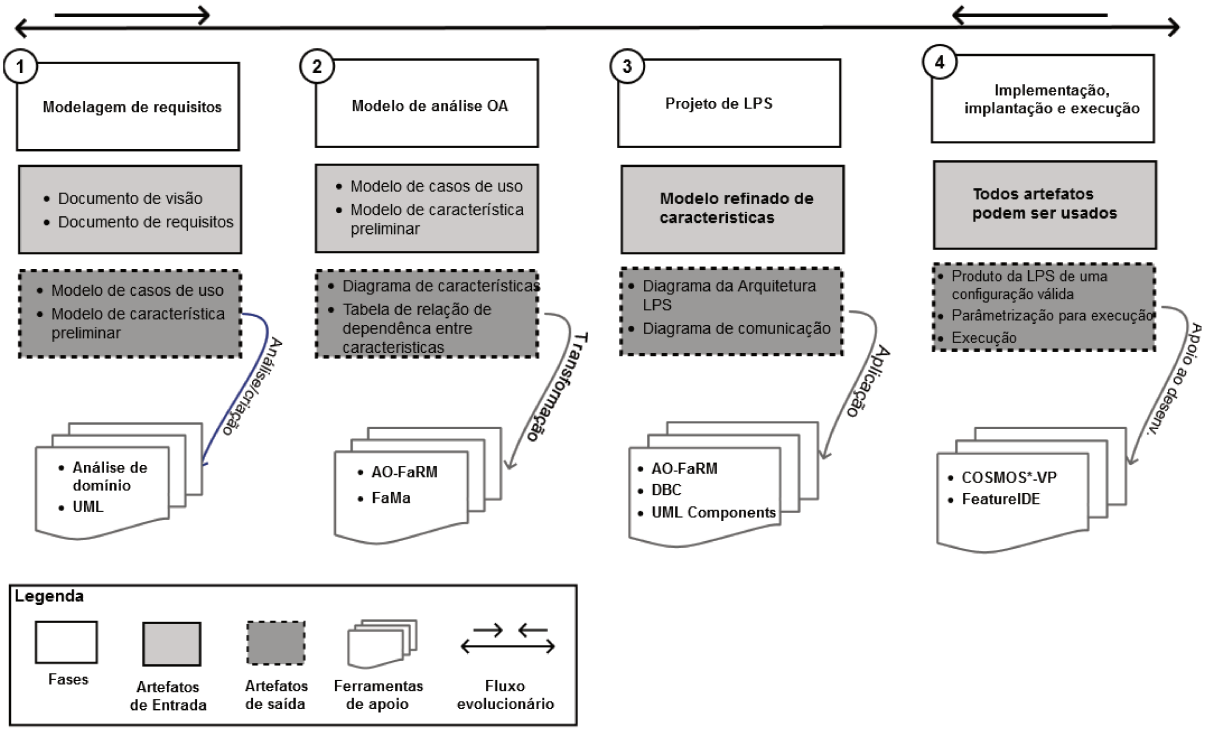



Figura 4.1: Fases da aplicação do Método PLUS

Esta análise também ajuda no entendimento ao identificar requisitos funcionais e não funcionais que são levantados. A saída desta fase alimenta a segunda fase, com o modelo de casos de uso e um modelo de características preliminar que foram criados a partir da visão, a definição de escopo abordado no documento de visão e os requisitos que atendem ao escopo.

### 4.1.2 Fase 2 – Modelo de análise OA

Nesta fase, o diagrama de característica e tabelas da relação das características, demonstra a dependência entre duas ou mais características. Ainda na fase 2, o mapeamento e transformação de características na visão de elementos arquiteturais já é aplicado adiantando um pouco da fase 3. Para tanto, são considerados a aplicação dos métodos FArM, AO-FArM e uma consequente validação do modelo de característica utilizando a FeatureIDE descrita na Seção 2.4.5. A aplicação dos métodos, colabora com um refinamento e permitirá detectar anomalias, erros e com possíveis adaptações necessárias.

Para aplicação do AO-FArM, uma análise utilizando o desenvolvimento orientado a aspectos foi realizada, e resulta em uma visão do modelo de característica Orientado a Aspectos (OA).

### 4.1.3 Fase 3 – Projeto LPS

Na fase 3, o modelo de característica já refinado é utilizado em uma transformação das características em elementos arquiteturais em forma de componentes. A aplicação do Desenvolvimento Baseado em Componentes (DBC) entra nesta fase, onde cada característica ou grupos de características são convertidos para componentes utilizando o COSMOS*-VP. Após a aplicação do COSMOS*-VP, tem-se a materialização das características em componentes.

### 4.1.4 Fase 4 – Implementação, implantação e execução

Por fim, na fase 4, a implementação ocorre a partir das diretrizes da engenharia de aplicação. Uma infraestrutura de LPS é criada utilizando todos os artefatos que foram armazenados em um repositório durante as fases anteriores. A partir desta infraestrutura, uma arquitetura de LPS é instanciada face a configuração de um produto. Quando gera-se o produto, as alternativas de projetos associadas a pontos de extensão da configuração são resolvidas, materializando um produto específico, neste caso, um monitor.

Por ser um ciclo evolucionário e iterativo, eventualmente foi necessário voltar nas fases iniciais e prosseguir a fim de evoluir o modelo de características e consequentemente evoluir os demais artefatos. Ao final da fase 4, tem-se uma configuração válida de um produto que possa ser implantado e executado.

## 4.2 FlexMonitorWS: Monitoração de Serviços Web flexível

A FlexMonitorWS é uma solução de monitoração criada a partir da adoção de LPS que tenta preencher lacunas identificadas nas soluções de monitoração de serviços Web. Estas lacunas foram identificadas a partir de uma Revisão Sistemática da Literatura vista no Capítulo 3.

### 4.2.1 Visão geral da FlexMonitorWS

A FlexMonitorWS tem esta ideia de flexibilidade a partir de Perfis de monitoração ligado a taxonomia. Ambos foram descritos no Capítulo 2 nas Seções 2.2.1 e 2.2.2. Cada perfil atenderá a um objetivo específico, dadas as necessidades de um monitor a partir dos requisitos do usuário.

De acordo com a Figura 4.2, o monitor é gerado a partir deste perfil e será feita a instalação em pontos de monitoração, como no provedor ou no consumidor. Uma vez criado e definido o monitor, este por sua vez, deverá ficar concentrado em um repositório

de monitores em um Servidor. Os dados de monitoração resultantes do monitor são colhidos e sincronizados com o servidor.

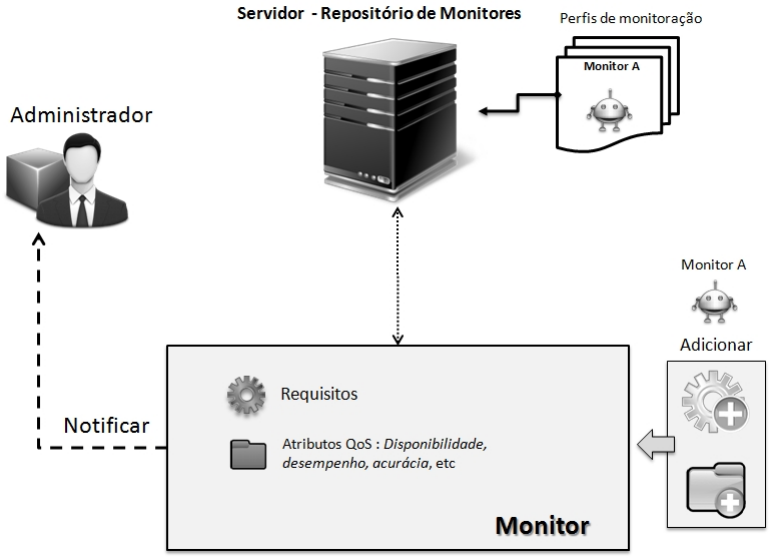



Figura 4.2: Visão da solução de monitoração

Uma vez que configurações de perfis de monitoração estejam confirmadas, serão sincronizadas com o Servidor, conforme representado na Figura 4.3. Desta maneira, o sincronismo ocorrerá tanto em nível de configuração dos perfis quanto em nível de coleta de dados. Os usuários analisarão os dados da monitoração por meio de uma interface. Os monitores também emitirão alarmes e notificação via arquivos de Log ou por e-mail.

### 4.2.2 FlexMonitorWS: Escopo da solução

O escopo da solução contempla um conjunto de funcionalidades fundamentais e são:

1. **Monitoração sobre Serviços Web para obter atributos de QoS** - Esta função permite efetuar um acompanhamento de atributos de QoS ao monitorar um Serviço Web. A solução deveria acomodar diferentes atributos de QoS com viés de flexibilidade ao permitir criar novos monitores com capacidade de atender aos atributos de QoS necessários.

2. **Monitoração sobre recursos de TI que podem degradar ou gerar algum impacto sobre o atributo de QoS** - Esta função permite identificar pontos de falhas no contexto onde o serviço é hospedado, como na aplicação servidora, servidor e na rede. É possível obter um entendimento da degradação de um dado atributo de QoS ao perceber que a aplicação servidora registra um Log de falhas contendo informações da execução do serviço Web. O servidor por sua vez pode ter oscilações

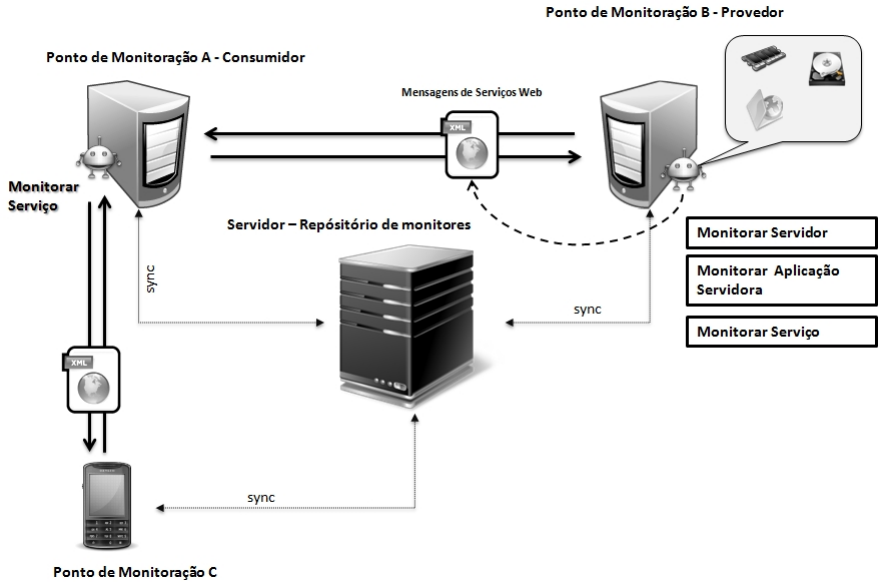



Figura 4.3: Coleta de dados e sincronismo de dados e configurações com o servidor

na memória quando seu uso não é exclusivo ao serviço Web. Enquanto que o acesso ao Servidor usando um Endereço IP e Porta, pode sofrer um baixo desempenho por alguma oscilação na rede.

3. **Monitoração abrangente no contexto geral** - A partir dos itens 1 e 2, a solução permite criar uma monitoração mais abrangente tendo de um lado, um monitor que acompanha atributos de QoS desejados sobre um serviço, e por outro lado, um conjunto de monitores cada um especificamente acompanhando os demais recursos, como aplicação servidora, servidor e rede.

4. **Flexibilidade** - Para criar uma solução flexível considera-se os aspectos que a solução deveria contemplar, visando permitir deixar pontos de extensão bem definidos. Fornecer um meio adequado que a solução possa permitir que novas funcionalidades sejam incluídas e que uma nova forma de monitoração seja tangível. Oferecer interfaces para atributos de QoS que não são escopos desta versão pode ser uma alternativa a ser considerada.

### 4.2.3 FlexMonitorWS: Requisitos funcionais

São requisitos funcionais da FlexMonitorWS:

**RF01** - Perfis de Monitoração

Deverá permitir a criação de Perfis de Monitoração. Este perfil deverá considerar um conjunto de atributos de QoS, ou considerar a monitoração sobre recursos como aplicação servidora, servidor ou rede.

**RF02** - Atributos de QoS

Para perfil de monitoração de serviços Web com objetivos de colher informações sobre atributos de QoS, são considerados os seguintes atributos de QoS:1) Desempenho; 2) Disponibilidade; 3) Robustez; 4) Acurácia; 5) Confiabilidade

**RF03** - Monitoração sobre contexto geral

Para a monitoração geral sobre o contexto, o sistema deverá permitir que o administrador selecione pontos de monitoração. São alvos que o administrador poderá selecionar:

1. Monitoração do Servidor - Efetuar a monitoração de memória e disco, no servidor onde hospeda o serviço web.
2. Monitoração da Aplicação Servidora - Efetuar a monitoração por meio da inspeção de arquivos de log da aplicação Servidora. Um ou mais arquivos poderão ser inspecionados.
3. Monitoração da Rede - Efetuar a monitoração da Rede por meio da verificação de Endereço IP/Porta destino

**RF04** - Controle do modo de operação do monitor

A solução deve monitorar respeitando o grau invasivo desejado pelo *stakeholder* interessado na monitoração. Ou seja, caso a monitoração seja transparente, o uso de interceptação de mensagens pode ser uma saída. Caso a frequência de execução da monitoração seja continua ou intermitente, modos de operação como invocação ou inspeção é uma alternativa.

**RF05** - Controle da frequência de execução

A solução deve permitir a configuração dos tempos de execução da monitoração.

**RF06** - Emissão de Notificação

A solução deve emitir algum tipo de notificação da monitoração realizada seja por email ou por arquivo de Log.

### 4.2.4 FlexMonitorWS: Requisitos Não Funcionais

São requisitos não funcionais da solução:

**RNF01** - Solução de monitoração flexível

É importante que a aplicação tenha flexibilidade em adicionar e remover atributos de QoS, mesmo que de forma estática.

**RNF02** - Autonomia na execução

A solução deverá executar com certa autonomia a monitoração do serviço, sem muita intervenção do administrador respeitando os intervalos que ele estabelece como monitoração, sendo uma frequência contínua agendada ou intermitente ou somente quando o serviço for usado.

**RNF03** - Efeitos colaterais de monitoração

A solução poderá se comportar de forma não intrusiva, intrusiva e levemente intrusiva, desde que a monitoração não traga efeitos colaterais ao serviço ou ao servidor onde a aplicação ficará em execução.

**RNF04** - Consumo de informações

A solução deverá fornecer meios de acesso ao aplicativo por meio do acesso Web.

**RNF05** - Segurança

A solução deve manter os dados sob segurança permitindo acesso a aplicação com usuário/senha.

**RNF06** - Estatísticas

A aplicação deverá fornecer relatórios sobre sua execução de forma transparente.

**RNF07** - Robustez

A aplicação terá que garantir um mínimo de robustez na execução das ações.

**RNF08** - Persistência

Os dados deverão ser gravados em disco.

## 4.3 FlexMonitorWS: Criação da LPS

Os artefatos resultantes da aplicação do Método Plus compõem o repositório de artefatos da LPS gerado para a FlexMonitorWS. Por utilizar o método pró-ativo, todos os artefatos foram criados a partir do zero.

As etapas utilizadas para esta fase são apresentadas na Figura 4.4.

De acordo com a Figura 4.4, do lado esquerdo tem-se os artefatos para entendimento e análise de domínio contemplada por uma Revisão Sistemática da Literatura e contribuiu com uma visão da solução proposta, bem como os requisitos. Do lado direito, os artefatos resultantes como casos de uso e modelo de características preliminar, foram guiados por meio de um processo pivô, sendo este processo a criação da taxonomia que permitiu identificar lacunas nas soluções existentes.

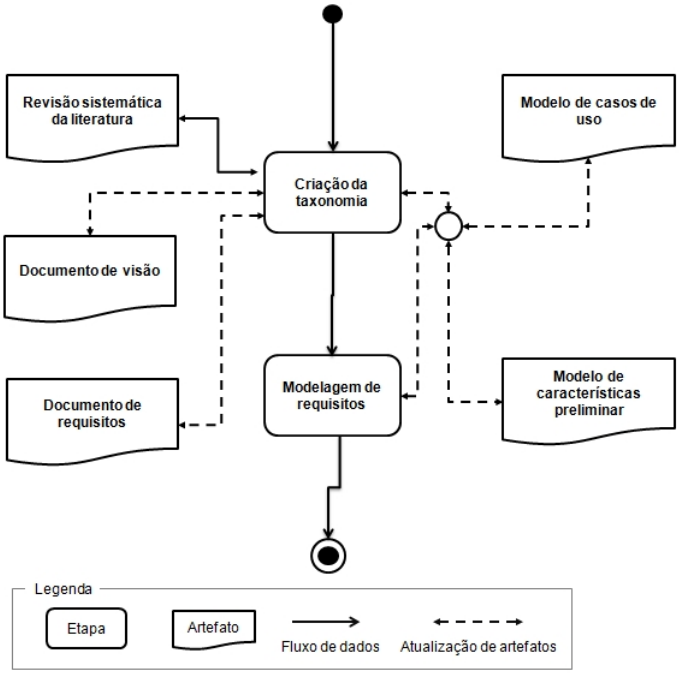



Figura 4.4: Etapas da fase inicial – Modelagem de requisitos

### 4.3.1 Fase 1: Modelagem de requisitos

Todo direcionamento aplicado nesta primeira fase foi orientado a aspectos e a características, seguindo o método PLUS e as orientações propostas por Tizzei [Tizzei12b]. As fases posteriores dependiam deste tipo de modelagem, uma vez que foi aplicado o método AO-FArM (Seção 2.7.2).

Esta fase tem na sua entrada o documento de visão e o documento de requisitos ambos gerados a partir da Análise de Domínio. Tais documentos apoiam a criação do modelo de casos de uso orientado a aspectos e um modelo de característica preliminar. O documento de visão forneceu um entendimento macro e o escopo definido da solução. Enquanto que os requisitos, foram minuciosamente elencados para este escopo, para ampliar o entendimento da solução.

#### Fase 1.1: Modelo de casos de uso orientado a Aspectos

Cada caso de uso tem uma relação com os requisitos funcionais descritos na Seção 4.2.3 deste capítulo.

O modelo de casos de uso é representado na Figura 4.5 através do Diagrama de casos de uso com a notação de Gomaa (2004) (descrito na Seção 2.5.1) utilizando o método PLUS. Note na Figura 4.5, os casos de uso são agrupados no diagrama para dar uma visão semelhante a fornecida pela taxonomia. Os pontos de extensão descritos no diagrama ajudam a entender onde ocorrerá uma possível variação e será tratada com uma

variabilidade identificada no modelo.

Os casos de uso foram agrupados da seguinte forma:

1. **Modo de operação e alvo**

   Agrupa casos de uso dos modos de operação da monitoração (interceptação, invocação e inspeção) e os possíveis alvos (Serviço, Aplicação Servidora, Servidor e Rede). Este agrupamento tem um caso de uso como núcleo, o Monitorar e se deriva em pontos de extensão em recursos (alvos). Sendo cada recurso, um novo caso de uso, como Serviço, Aplicação Servidora, Servidor e Rede.

2. **Atributos de QoS**

   Agrupa os casos de uso que são necessários para obtenção de valores de atributos de QoS. Este agrupamento tem casos de uso como núcleo Calcular Atributo de QoS e se deriva em pontos de extensão chamados de AQoS, podendo ser qualquer caso de uso representado neste agrupamento.

3. **Frequência de execução**

   Contém dois casos de uso distintos, onde ambos são comuns a toda a solução, uma vez que a frequência da monitoração terá sempre uma determinada periodicidade.

4. **Notificação**

   Agrupamento que representa os possíveis tipos de notificação da solução. O agrupamento tem casos de uso como núcleo Notificar e que se deriva em pontos de extensão dos tipos de notificação, sendo Enviar mensagem e Gerar Log.

5. **Controle e gerenciamento**

   Este agrupamento é composto por casos de uso que facilitará o controle e gerenciamento da solução.

Note também na Figura 4.5, os casos de uso utilizam uma notação específica. Os Pontos de extensão são definidos com a notação (*extensions points*) e carregam uma guarda. Esta guarda determina uma informação relacionada a variabilidade. Como por exemplo, no caso uso *Monitorar*, o ponto de extensão tem o nome de recurso determinando o tipo de recurso o qual seria monitorado, podendo ser algum dos casos de uso que estendem *Monitorar*.

### Modelo de característica preliminar

Com base nos artefatos gerados até então, como Modelo de Casos de Uso, o documento de visão e o documento de requisitos, temos o escopo definido para a FlexMonitorWS. A partir destes artefatos foi possível projetar um modelo de característica preliminar e foi

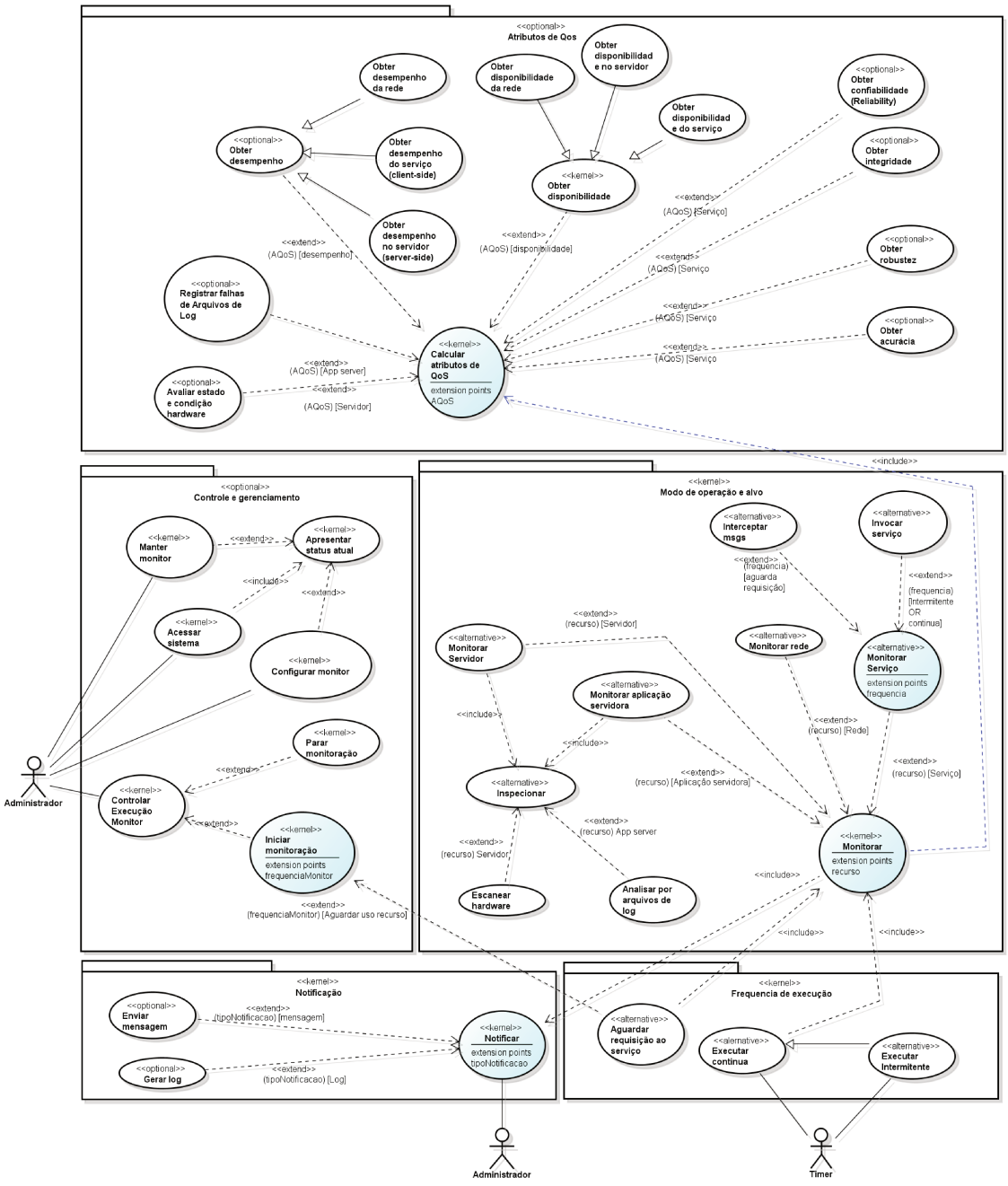



Figura 4.5: Diagrama de casos de uso da FlexMonitorWS

refinado nas seções posteriores. Na Figura 4.6 é representado o modelo, e onde também foi usado a notação proposta por Gomaa (2004). As regras de composições de características interdependentes do modelo são:

1. Aplicação Servidora requer o Atributos de QoS → Falhas em arquivos de Log E Modo de Operação → Inspeção.
2. Serviço requer ao menos um Atributo de QoS E Modo de Operação → Invocação OU Interceptação
3. Rede requer Atributo de QoS → Desempenho OU Disponibilidade
4. Interceptação requer Frequência de Execução → Aguardar requisição
5. Servidor requer Modo de Operação → Inspeção E Atributo de QoS → Estado e condição do Hardware

O modelo de característica representado no diagrama da Figura 4.6 tem relação de um-para-um com os elementos da taxonomia vista na Seção 2.2.1. Nesta fase, cada característica identificada e que pertence a taxonomia poderia ser representada como característica isolada, ao invés de agrupá-las.

### 4.3.2 Fase 2: Modelo de Análise OA

A análise de requisitos ocorreu primeiramente orientada por aspectos e características seguindo os passos descritos por Tizzei (2012b). A aplicação do AO-FArM contribuiu com a transformação das características em elementos arquiteturais e na separação de interesses base e transversais.

#### Fase 2.1: Método AO-FArM

De acordo com a Seção 2.7.2 para aplicação do AO-FArM é necessário obter a Visão de característica OA. Este modelo é obtido a partir de uma análise de requisitos orientada a aspectos e do modelo características, onde ocorre uma separação dos interesses base dos interesses transversais).

#### Fase 2.1.1: Visão de característica OA

Anteriormente à aplicação do AO-FArM é necessário executar algumas atividades relacionadas a análise. A primeira delas é identificar e compor interesses transversais. É usado como artefato de entrada, a especificação de requisitos e o modelo de característica e produz a matriz que relaciona características e interesses descrita na Tabela 4.1. Onde, uma característica folha é afetada por um interesse transversal, seus ancestrais também são afetados porque o modelo de característica é hierárquico.

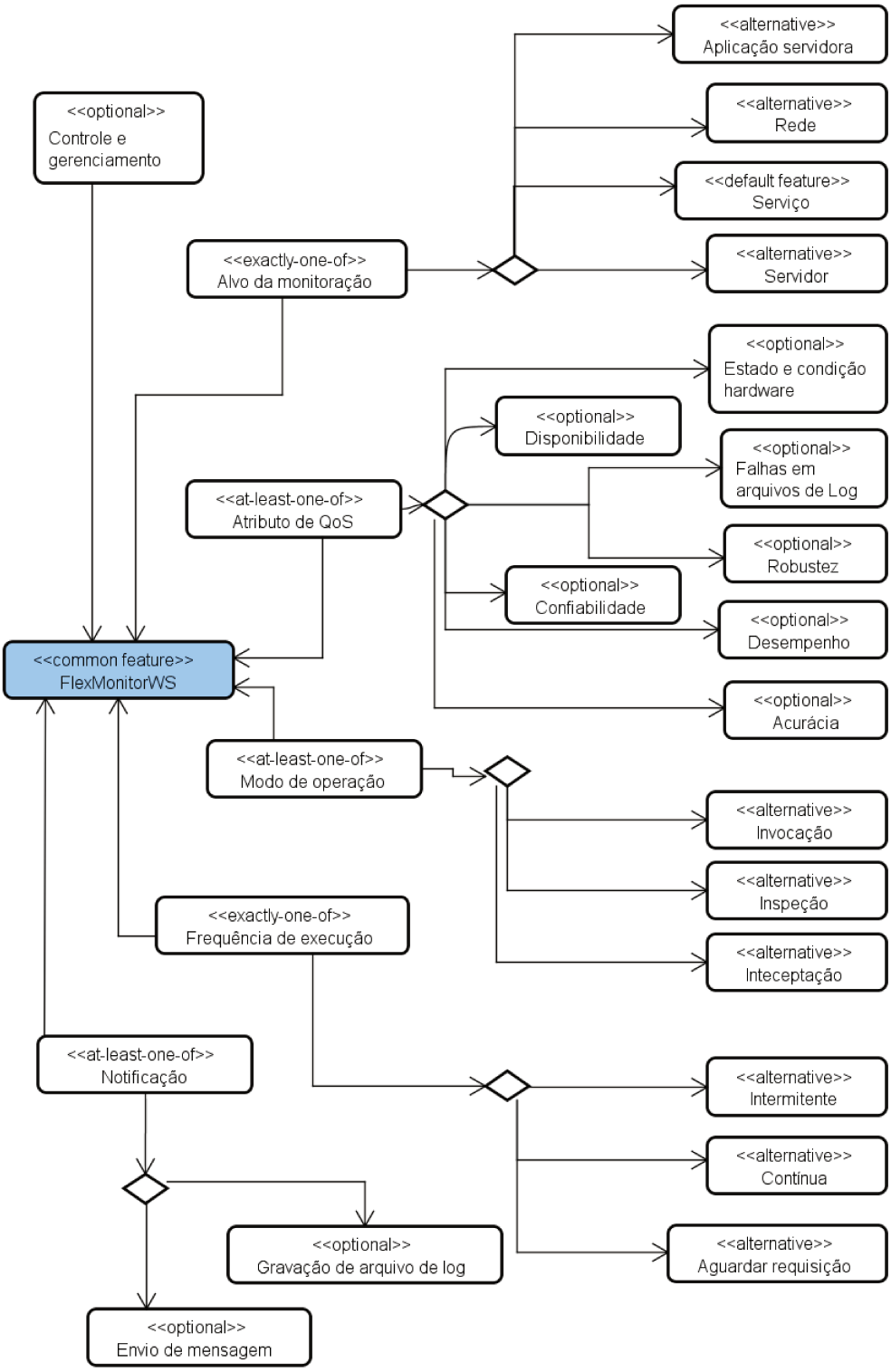



Figura 4.6: Diagrama preliminar do Modelo de Características

Tabela 4.1: Matriz relacionando característica de interesses transversais e características base

| **Característica base** | **Interesses transversais** | |
|---|---|---|
| | Relatório | Tratamento de exceção |
| Serviço | • | |
| Servidor | • | |
| Aplicação Servidora | • | |
| Rede | • | |
| Disponibilidade | • | |
| Confiabilidade | • | |
| Robustez | • | |
| Acurácia | • | |
| Desempenho | • | |
| Falhas em arquivos de Log | • | |
| Estado e condição do hardware | • | |
| Atributo de QoS | | • |
| Alvo da monitoração | | • |
| Modo de operação | | • |
| Notificação | | • |

**Especificar Casos de Uso Base e Transversais com Variabilidade**

Ainda na identificação e composição da Visão de característica OA, é necessário especificar quais casos de uso representam interesses transversais e não-transversais, mantendo-os separados utilizando a notação original com os estereótipos obrigatório, opcional e alternativo. Esta separação visa a enfatizar a variabilidade no modelo de casos de uso.

A Figura 4.7 mostra parte do diagrama de casos de uso da LPS da solução FlexMonitorWS. Observe que, alguns casos de uso foram omitidos por questão de espaço, o diagrama completo está representado na Figura 4.5. No diagrama representado pela Figura 4.7, os casos de uso de extensão como os atributos de QoS como por exemplo, *Obter disponibilidade, Obter desempenho* são casos de uso funcionais, portanto, casos de uso base. Estes casos de uso são ligados ao caso de uso transversal *Gerar resultados*.

Na Figura 4.8 temos o interesse transversal *Tratar Exceção*. Note que somente os casos de uso base com pontos de extensão conterão tratamento de exceção. Isso porque os demais casos de uso deverão gerar uma exceção na pilha de exceções, de forma a elevar o nível da exceção acima. Ou seja, para quem efetuou a chamada. Por exemplo, se caso ocorrer uma exceção ao calcular o atributo de QoS disponibilidade (exemplo em Java seria *throws Exception*), a exceção será enviada para a chamada anterior Calcular atributo de QoS, e ali será executado o caso de uso transversal *Tratar Exceção*.

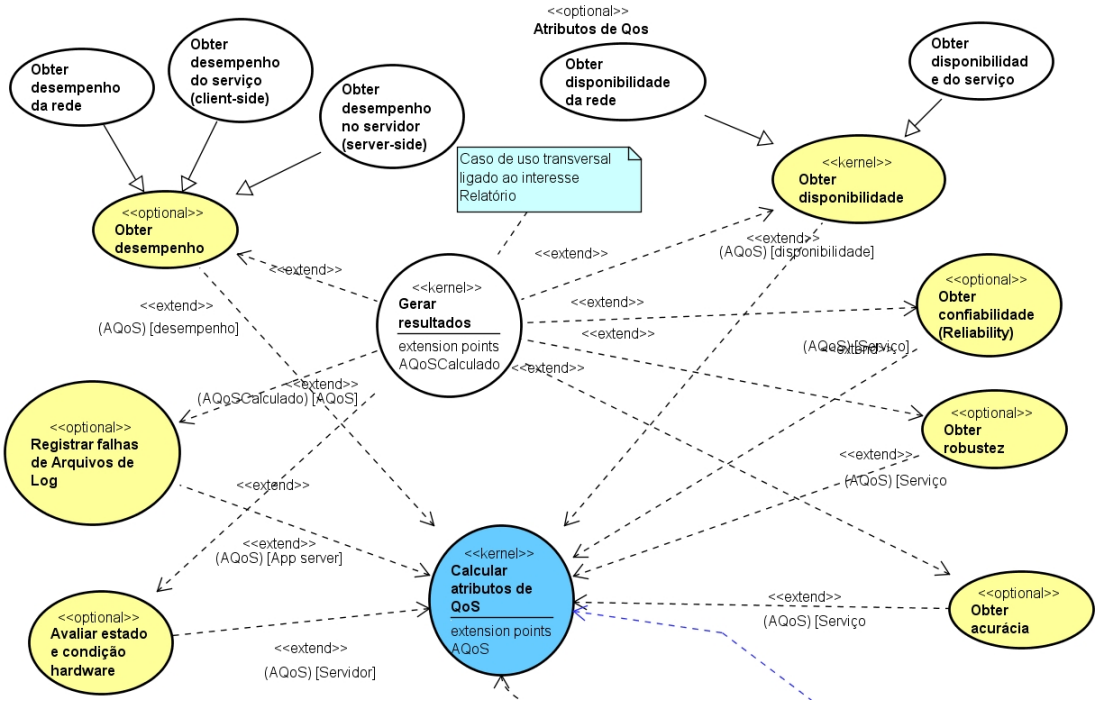



Figura 4.7: Especificação de interesses transversais e variabilidade no diagrama de casos de uso – Interesse transversal Relatório

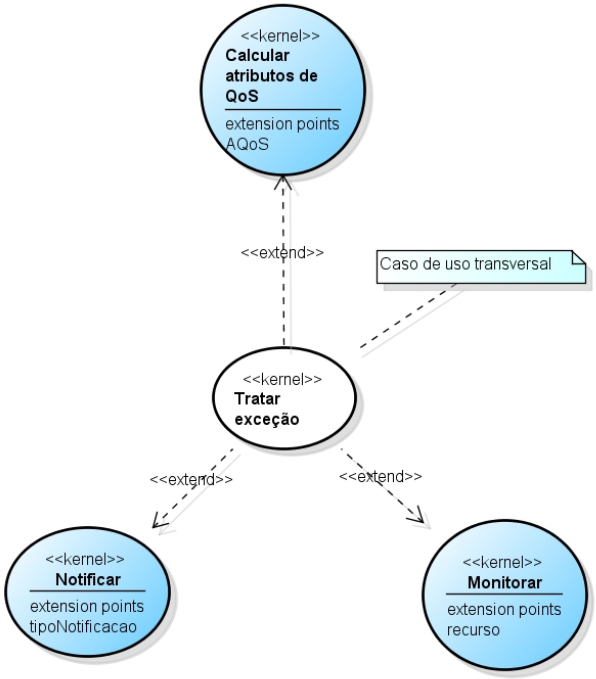



Figura 4.8: Especificação de interesses transversais e variabilidade no diagrama de casos de uso – Interesse transversal Tratar exceção

**Associar Características com Casos de Uso**

Nesta fase de modelagem da Visão de característica AO, há uma atividade que já é proposta no método PLUS. Segundo Tizzei (2012b), esta atividade de associação de característica com caso de uso visa facilitar a propagação de mudanças por meio do rastreamento, os modelos de características e de casos de uso são complementares. O modelo de característica especifica de forma gráfica, estática e hierárquica as características com foco na variabilidade, já o modelo de casos de uso, especifica a interação entre usuário e o sistema. Se um dado caso de uso estende ou inclui outro e os dois foram mapeados para características diferentes, então existe uma dependência entre as características.

A Tabela 4.2 representa esta associação. Na primeira coluna são listados os grupos de características; Na segunda coluna a categoria do grupo. A terceira coluna reflete a categoria interna ao grupo de características. Na quarta coluna, quais casos de uso estão relacionados com o grupo de características do modelo de características; Na quinta coluna, a categoria do caso de uso, e por fim, na última coluna, o nome do ponto de variação que está ligado ao ponto de extensão demonstrado no diagrama de caso de uso.

Ainda na Tabela 4.2, note os dois últimos casos de uso, são os casos de uso transversais e o nome do ponto de variação é exatamente a informação a ser capturada em tempo de execução. Para Gerar resultados é necessário haver o atributo de QoS calculado. Enquanto que para Tratar uma exceção é necessário que ela ocorra.

A última fase é Analisar Características Base e Transversais, cujo objetivo é permitir a análise racional da influência das características transversais nas demais características e ocorre nas matrizes criadas até então, no modelo de característica e nos casos de uso que representam casos de uso base e transversal. Como resultado dessa atividade, tem-se a visão de características OA. A aplicação dos passos a seguir, foram descritos na Seção 2.7.2.

No passo 1, as características transversais identificadas como representantes de interesses transversais foram separadas.

No passo 2, as características que são entrecortadas por outras características foram representadas com apoio da Tabela 4.4 que representa a matriz que relaciona características e interesses.

No passo 3, a resolução de conflitos onde três características entrecortavam uma outra característica, aplicou-se a notação de precedida-por.

A Figura 4.9 mostra o resultado parcial da Visão de características OA após a aplicação dos passos. As características transversais *Relatório e Tratamento de Exceção* são representadas na Figura 4.9. A característica *Relatório* entrecorta todos os atributos de QoS que são alternativas que a solução oferece. Enquanto que, a característica *Tratamento de Exceção* entrecorta todas as características base obrigatórias que são anteriores às características folha do modelo.

Tabela 4.2: Relação de dependência entre características e casos de uso

| Grupo de características | Categoria do grupo | Categoria | Caso de Uso | Categoria Caso de Uso | Nome do ponto de variação |
|---|---|---|---|---|---|
| **Alvo da monitoração** | exactly-one-of | Common | Monitorar | Kernel | NA |
| | | Alternative | Monitorar Serviço | Alternative | Recurso |
| | | Alternative | Monitorar Aplicação servidora | Alternative | Recurso |
| | | Alternative | Monitorar Servidor | Alternative | Recurso |
| | | Alternative | Monitorar | Alternative | Recurso |
| **Atributos de QoS** | at-least-one-of | Common | Calcular atributo de QoS | Kernel | NA |
| | | | Obter Disponibilidade | Optional | AQoS |
| | | | Obter Confiabilidade | Optional | AQoS |
| | | | Obter Acurácia | Optional | AQoS |
| | | | Obter Desempenho | Optional | AQoS |
| | | | Obter Robustez | Optional | AQoS |
| | | | Registrar falhas de arquivos de Log | Optional | AQoS |
| | | | Avaliar estado e condição do hardware | Optional | AQoS |
| **Modo de operação** | at-least-one-of | Common | Monitorar | Kernel | NA |
| | | | Invocar serviço | Alternative | Frequência |
| | | | Interceptar mensagens | Alternative | Frequência |
| | | | Inspecionar | Alternative | Frequência |
| **Frequência de execução** | exactly-one-of | Common | Monitorar | Kernel | NA |
| | | | Execução contínua | Alternative | NA |
| | | | Execução Intermitente | Ponto de variação | NA |
| **Notificação** | at-least-one-of | Common | Notificar | Kernel | NA |
| | | | Enviar mensagem | Optional | Tipo Notificação |
| | | | Gravar arquivo de Log | Optional | Tipo Notificação |
| **Relatório** | exacIty-one-of | Kernel | Gerar resultados | Kernel | AQoS Calculado |
| **Tratamento de exceções** | exacIty-one-of | Kernel | Tratar exceção | Kernel | Exceção |

### Fase 2.2.2: Aplicação do AO-FArM

É necessário aplicar a primeira etapa de AO-FArM que consiste de remover características não relacionadas à arquitetura e resolver características de qualidade.

1. A primeira etapa é Remover características não-relacionadas à arquitetura e resolver características de qualidade:

   Durante esta etapa, percebemos que a característica Controle e Gerenciamento não possuía nenhuma relação com as demais características adjacentes, nem com as subjacentes, além de ter uma arquitetura não-relacionada a solução e foi removida do modelo. Porém, a característica em questão compõe um conjunto de funcionalidades que serão implementadas à parte da solução principal. Esta característica é a GUI (sigla do inglês *Graphics Unit Interface*) interface gráfica de operação da solução.

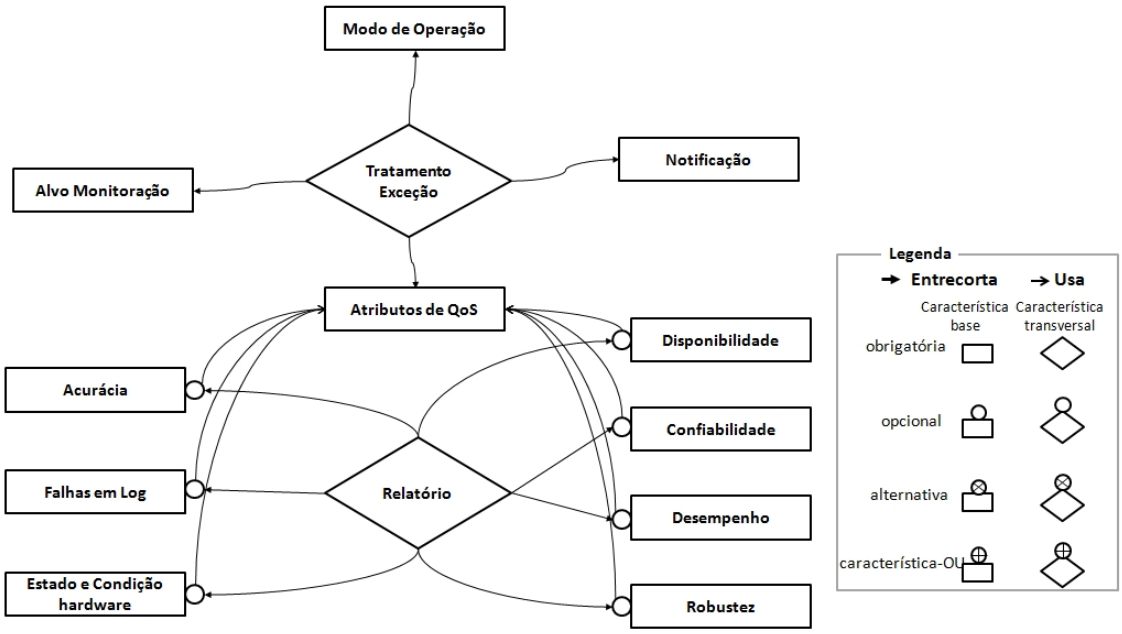



Figura 4.9: Visão de características OA

2. A segunda etapa é obtida ao Transformar baseando-se nos requisitos da arquitetura:

   Como a arquitetura do projeto composta por muitos protocolos ligados ao Serviço Web, cada protocolo seria uma característica e tornaria a abstração da solução impertinente. Portanto, a análise e aplicação desta etapa no modelo de característica não modificou o modelo.

3. A aplicação da penúltima etapa Transformar baseando-se nas relações de interação entre características:

   A característica *Aguardar requisição* subjacente a *Frequência de execução* contradiz *Interceptação*, uma vez que a característica *Interceptação* contém a regra de negócio para efetuar algum controle muito semelhante à característica *Aguardar requisição*, sendo uma contradição identificada no modelo e, portanto, característica removida do modelo.

A primeira, segunda e terceira etapas propostas pelo método AO-FArM uma vez concluídas, resultou em um modelo de característica refinado e transformado. Além disso, a aplicação destas etapas acima mencionadas, foi possível a obtenção da visão do modelo de característica transformado representado na Figura 4.10.

As demais etapas restantes do método AO-FArM consistem da transformação de características ou grupo de características em elementos arquiteturais e será visto na próxima fase.

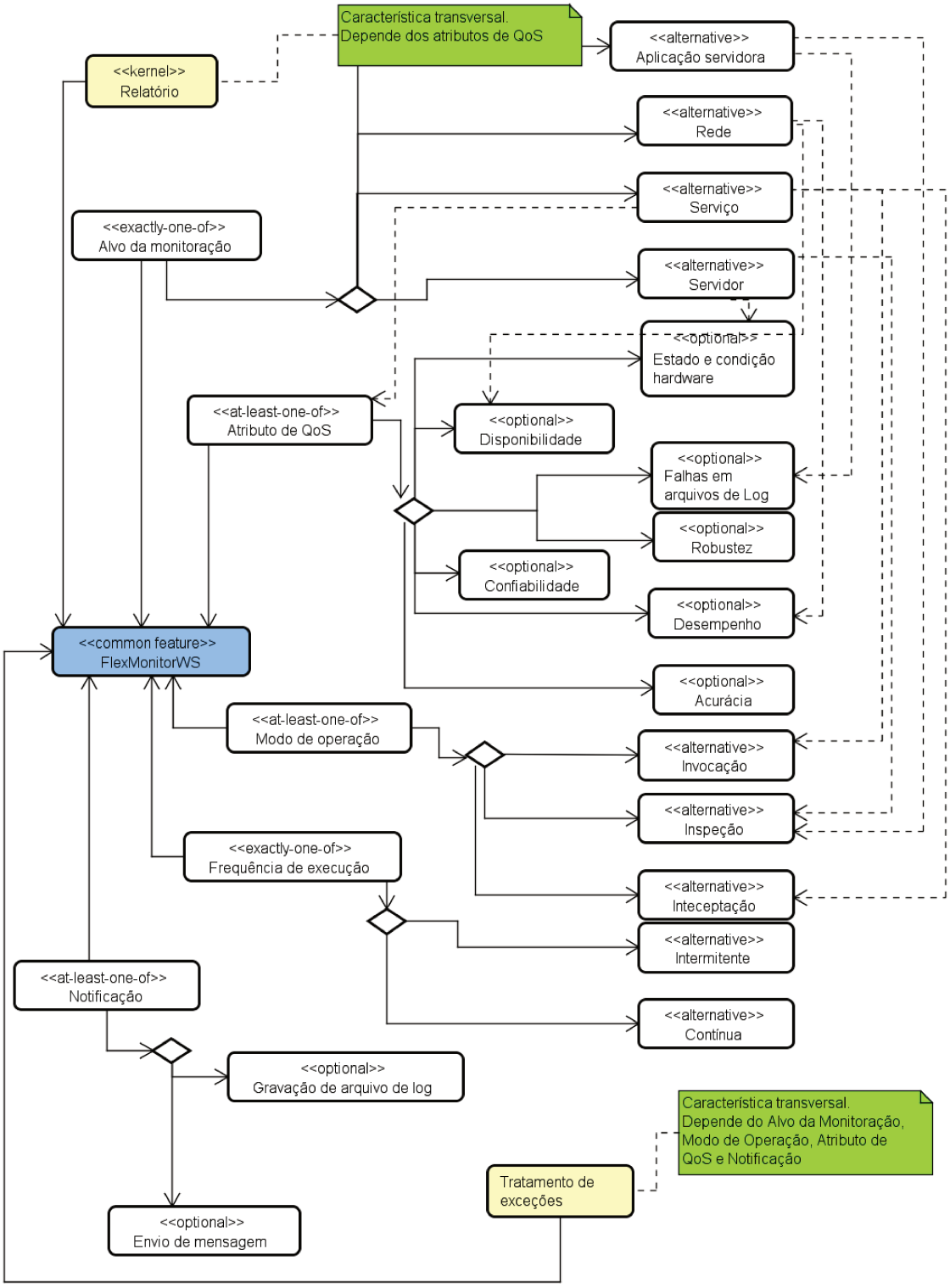



Figura 4.10: Modelo de Características refinado e transformado

### 4.3.3 Fase 3: Projeto de LPS

Esta fase inicia-se com a aplicação da última etapa do AO-FArM e é concluída com a aplicação do COSMOS*-VP. As etapas são orientadas no uso do desenvolvimento baseado em componentes e faz uso de diagramas de comunicação da UML.

#### Fase 3.1: Última etapa da aplicação AO-FArM

Nesta última etapa, convertemos características para componentes isolados para algumas das características que são alternativas, e as demais características opcionais convertidas para grupos de componentes com uma granularidade maior. Como por exemplo, no caso de *Atributos de QoS*, todos os atributos de QoS onde a regra de composição é dada mediante a uma característica adjacente como *Alvo da monitoração e Modo de operação*, fará parte de um único componente. Isto é, quando o *Alvo da monitoração* requer atributos de Estado e Condição como *Falhas em arquivos de Log, e Estado e condição do hardware*, tais atributos são agrupados em um único componente. Atributos relacionados a eventos temporais como *desempenho, disponibilidade* e os demais, serão agrupados em *Atributos de Eventos temporais*.

A característica *Notificação* é uma generalização dos tipos de notificações e, portanto, será um componente isolado.

A característica *Frequência de execução* se enquadra na relação onde suas subcaracteristicas como *Intermitente e Contínua* podem ser vistas como funcionalidades internas à supercaracterística. Portanto, configurando apenas um componente para a característica em questão.

A Figura 4.11 representa o diagrama da Arquitetura inicial de Linha de Produto. Através do diagrama, note a divisão dos componentes conforme descrito acima.

#### Fase 3.2: Refinamento da Arquitetura LPS baseada em componentes e aspectos

Após obter a Arquitetura inicial de LPS representado na Figura 4.11 através do Diagrama de ALP, é necessário especificar como eles se relacionam, especificando interfaces e conectores. A materialização da arquitetura baseada em componentes e aspectos é realizada através de COSMOS*-VP.

#### Fase 3.2.1 Identificar interfaces bases e transversais

Seguindo as orientações segundo Tizzei (2012b), para aplicação desta fase é necessário alguns artefatos gerados até então, como casos de uso base e transversais com variabilidade, os modelos de características transformados e a visão de característica OA transformada.

A Figura 4.12 mostra a arquitetura LPS baseada em componentes e aspectos utilizando AO-FArM. Repare na Figura 4.12, componentes que contêm na hierarquia a integração

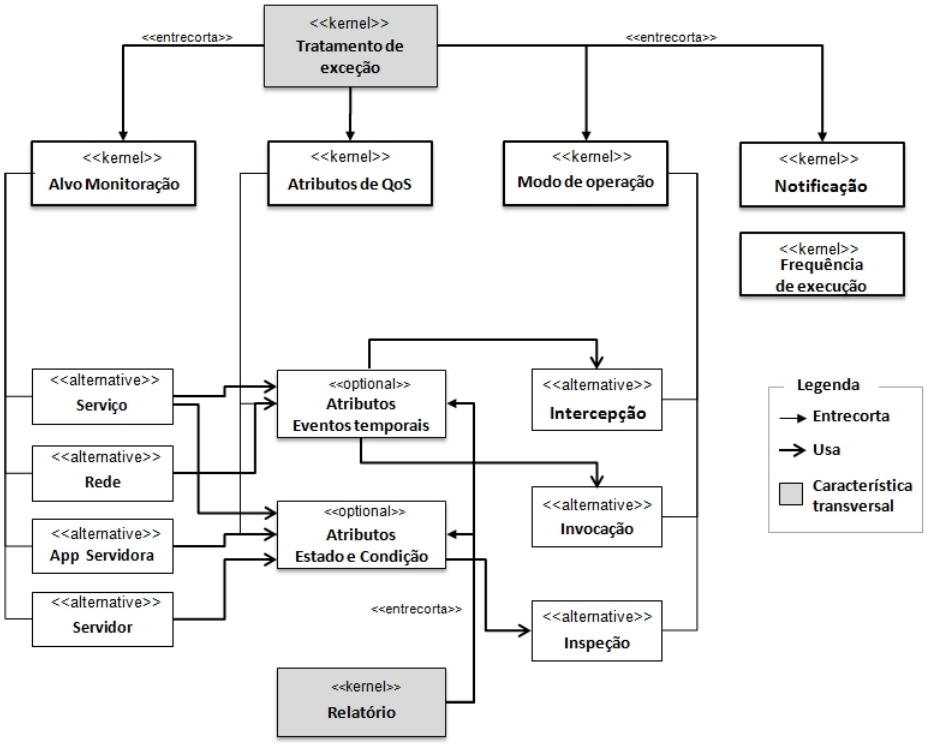



Figura 4.11: Diagrama da Arquitetura inicial de Linha de Produto

com outros componentes transversais na sua composição, contém uma notação sendo um losango interligando um componente base com um componente transversal (Tizzei12b).

### Fase 3.2.2: Especificar as operações das interfaces bases e transversais

Após a identificação das interfaces bases e transversais de cada componente, as interações entre os componentes devem ser especificadas através do diagrama de comunicação. Primeiramente, são especificadas as interações entre as interfaces base providas e requeridas. Para dar apoio ao entendimento dos métodos das interfaces e das assinaturas destes métodos, o modelo de casos de uso orientado a aspectos é utilizado (Fase 1 e Fase 2). Essa atividade é similar a atividade de especificar interfaces usada em DBC descritos na Seção 2.6.

A Figura 4.13 representa a comunicação entre os componentes descritos na Arquitetura de Linha de Produto. A ordem de comunicação define em que momento ocorre a ligação entre os componentes a partir das interfaces e dos métodos.

Ainda na Figura 4.13, a partir do componente central FlexMonitorWS é configurada a Frequência de Execução da monitoração por meio de um *Timer*, que uma vez acionado gera uma interrupção na máquina virtual e dispara o processo de monitoração. A etapa posterior ocorre no Modo de Operação que obtém amostras e informações sobre atributos de QoS por meio das maneiras associadas a este ponto (e.g. invocação, interceptação ou

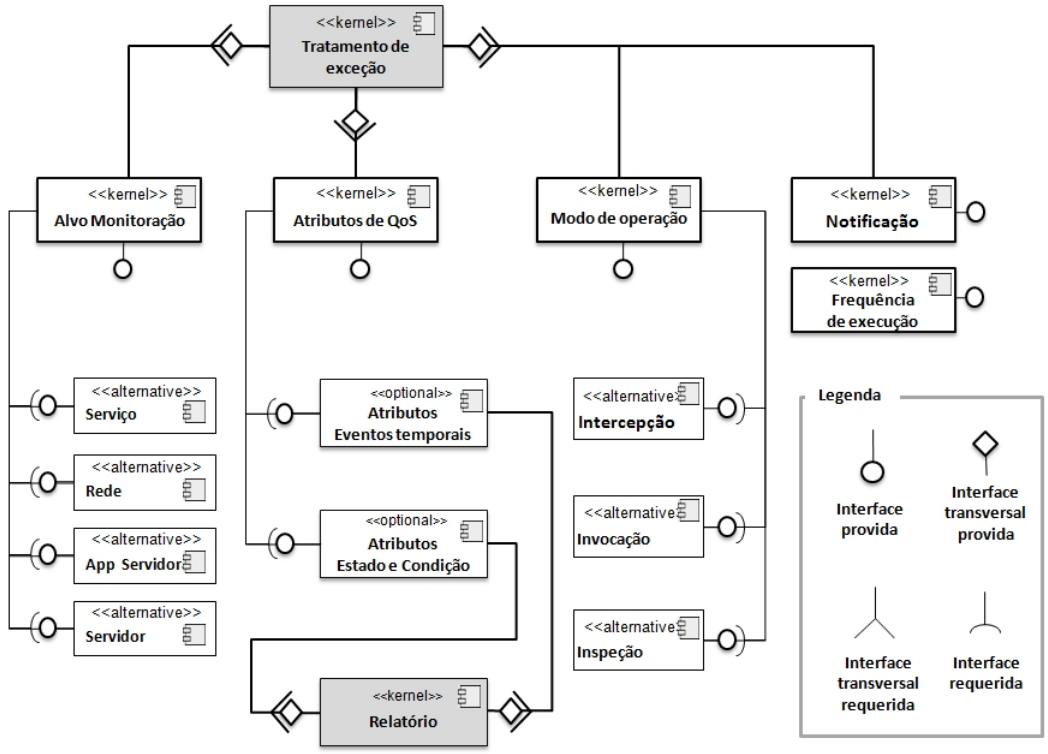



Figura 4.12: Arquitetura da FlexMonitorWS baseada em componentes e aspectos

inspeção). A etapa seguinte ocorre no objeto AQoS, onde os atributos de QoS associados a este ponto, são processados e calculados os valores para cada atributo. O objeto de Notificação finaliza o processo, tendo em Tipos de notificação (e.g. envio de email ou salvar arquivo de Log) as possíveis maneiras de notificar os interessados sobre os resultados obtidos durante a monitoração. Observe que na Figura **??** os elementos no diagrama que são do tipo *Control* e *Entity* permitem oferecer uma visão ligada a pontos de extensão da solução e as possíveis alternativas de projeto associadas a estes pontos.

As demais etapas sugeridas por Tizzei (2012b) seria a aplicação da análise de componentes legados. Como aqui os artefatos foram criados do zero aplicando a metodologia proativa, não se faz necessário o emprego desta etapa, cabendo agora a implementação da LPS.

### 4.3.4 Fase 4: Implementação, implantação e execução

Nesta a aplicação do modelo COSMOS*-VP descrito na Seção 2.8.2 se fez necessária. Este modelo fornece diretrizes para codificar uma arquitetura LPS baseada em componentes e aspectos.

A Figura 4.14 mostra como é realizada a materialização dos componentes *Alvo* e *Serviço* ligados por um Conector-VP. Repare na Figura 4.14, o *AlvoAdapter* estende *AA-*

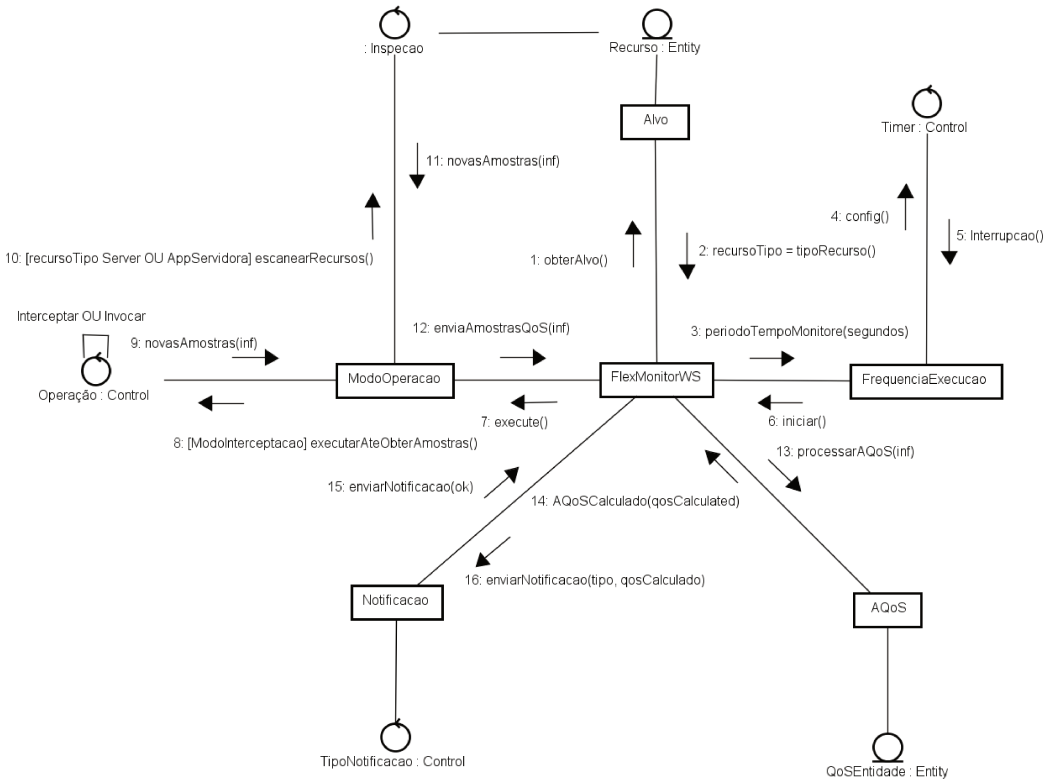



Figura 4.13: Diagrama de comunicação representando a Arquitetura LPS final baseada em componentes e aspectos

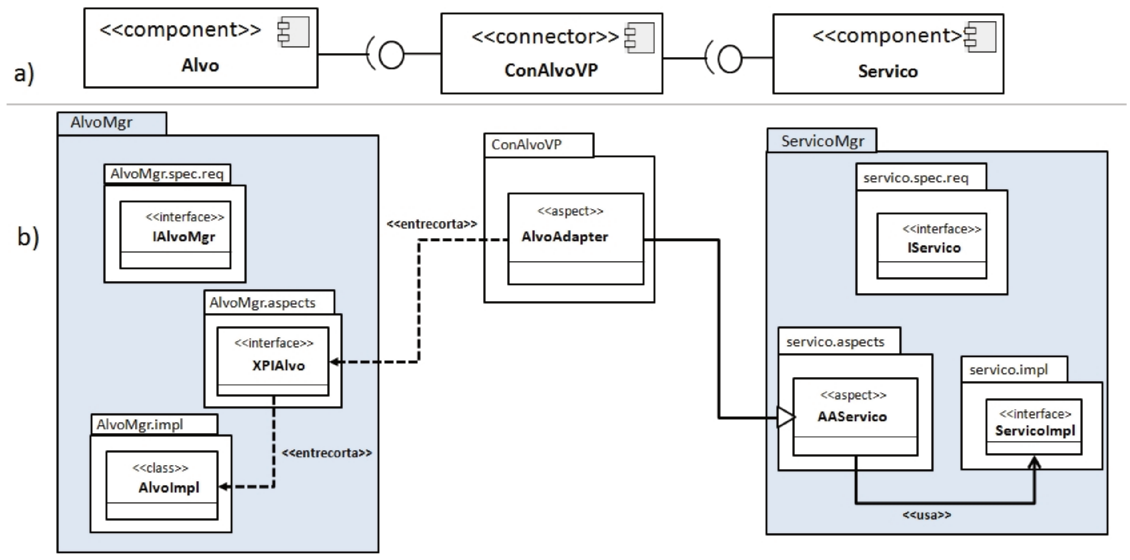



Figura 4.14: a) Visão arquitetural dos componentes base (Alvo e Serviço) integrados por um conector; b) projeto detalhado dos componentes e o conector usando o modelo COSMOS*-VP

*Servico* dessa forma o componente *ServicoMgr* não irá expor suas classes internas.

### Detalhes da implementação

Em alguns componentes da solução, optamos pela reutilização de bibliotecas ou APIs existentes em atendimento a um requisito funcional da solução. Como nos casos dos componentes de modos de operação, Invocação, Interceptação e Inspeção. No caso da Invocação, a biblioteca SAAJ (*SOAP with Attachments API for Java* – SOAP API com anexos para Java) [Jayanti05] é usada em sistema de mensagens SOAP que fornece uma maneira padrão de enviar documentos XML pela internet. A SAAJ é usada para manipular a mensagem SOAP para o contexto apropriado e os anexos as mensagens são montados em tempo de execução.

Ao derivar um XML de um serviço Web em partes, permite que as partes sejam montadas a partir de informações que o usuário fornece. No caso deste projeto, as partes que constituem o XML de uma mensagem de requisição SOAP ficam em um arquivo com extensão *properties* (arquivo de propriedades), como endereço de acesso ao XML que define os contratos e as operações e os parâmetros necessários de uma dada operação.

O **TCPDUMP** [Tcpdump013] foi utilizado para o modo de operação por interceptação. O TCPDUMP é um analisador de pacotes que são transmitidos pela rede em baixo nível. Ao executar o TCPDUMP com um conjunto de parâmetros que são passados a ele, é possível estabelecer filtros. Estes parâmetros são baseados em expressões e regras booleanas e permite capturar pacotes com um interesse específico. No caso deste projeto, o seguinte comando do TCPDUMP foi utilizado:

**tcpdump -avvlASs 0 dst or src host «ip»**

Ao executar o comando do TCPDUMP ele se manterá "escutando"o tráfego da rede e imprimir as saídas do que está sendo recebido/transmitido de acordo com o filtro estabelecido em console. A saída no console é obtida e analisada a partir de expressões regulares que determinam a existência de uma dada informação sobre o serviço Web. Como por exemplo, se existe uma informação de uma *WS-Policy* implementada no serviço Web ao analisar o XML que é transmitido pela rede.

A única desvantagem da forma como é utilizado o TCPDUMP neste projeto, é que não é possível detectar informações em protocolo HTTPS (SSL – *Secure Service Layer*), pois os dados são criptografados.

Para o modo de Inspeção no Servidor, foi utilizada uma biblioteca da *Hyperic Sigar* [Sigar10], é uma API com distribuição de código fonte aberto, e sendo também software livre. Esta API oferece uma interface portável para obtenção de informações sobre o sistema, como memória, memória *swap*, CPU, média de carga, tempo ligada, processo por memória, sistema de arquivos e uma variedade de informações.

No caso do componente Rede o comando *Ping* baseado no protocolo ICMP foi utilizado

para obter estatísticas de disponibilidade e desempenho da rede. Este comando está presente na maioria dos sistemas operacionais, sua saída no console é padrão o que permitiu obter os dados para análise utilizando o mesmo comando em diferentes plataformas. Pode haver necessidade de adaptação, pois nem todas as plataformas foram testadas, somente aquelas que estão presentes nos estudos de caso.

### Processo interno de injeção de falhas

Como dito na Seção 2.3.5 (Robustez), a injeção de falhas no serviço Web é por meio da modificação de uma requisição (*workload*) inserindo scripts que criam uma requisição de falha (*faultload*). Para tanto, os passos de criação da injeção é definida na Figura 4.15.

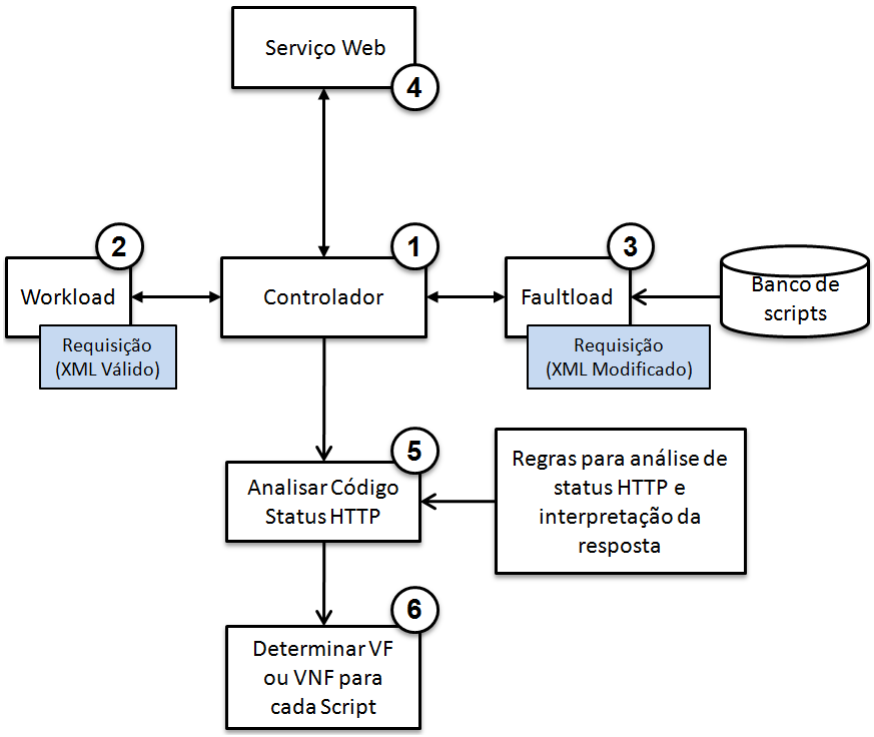



Figura 4.15: Processo interno da FlexMonitorWS de injeção de falhas

Os scripts podem ser configurados no arquivo de parâmetros denominado por target.properties e são incluídos no XML do serviço durante a execução de forma automática ou manual. Na execução automática, há um conjunto de 6 scripts que são criados e para cada serviço elencado para a monitoração do atributo de QoS robustez, serão injetadas as falhas. Ao final, um relatório será emitido a cada ciclo de monitoração.

De acordo com a Figura 4.15 as etapas para execução respeitam a seguinte ordem:

1. O Controlador obtém o XML de requisição para o preparo da injeção;

2. Como a primeira requisição ao serviço é realizada anteriormente a este processo para verificar a disponibilidade, esta requisição é mantida em memória e é a *workload* descrita na Figura 4.15;

3. A partir do target.properties ou dos scripts internos, é preparado um conjunto de XML criando diferentes *faultloads*;

4. Disparar contra o serviço cada *faultload*;

5. Analisar o código de status HTTP e verificar a partir do parâmetro definido no target.properties (service.robustness.script.expected.response) e compará-lo. Na maioria dos casos, se for 400 não há vulnerabilidade ao script malicioso, se for 500, a análise deve ser da mensagem que acompanha o código. Se for 200, possivelmente pode existir uma vulnerabilidade. Porém, é necessário descartar os falsos positivos, aqui pode ser necessária uma apuração de técnicos e administradores;

6. Se a análise é conclusiva, será possível determinar se foi encontrada uma vulnerabilidade (VF) ou não (VNF).

No arquivo target.properties para execução da injeção é obrigatório a existência do parâmetro service.robustness.script.auto igual a 1. A Figura 4.16, representa os parâmetros que devem ser embutidos para a execução de forma manual ou considerar a execução nos dois modos.

```
# Fault injection execution mode
# set robustness.auto = 1 generate 6 scripts automatic way
# set robustness.auto = 0 manual scripts must be created
# set robustness.auto = 2 both modes of execution are applied
service2.robustness.auto = 2

# Scripts manual execution
service2.robustness.script1.name = Manual 1 - Malformed XML
service2.robustness.script1.replace = </cardNumber>, <cardNumber>
service2.robustness.script1.expected.response = 400,500
```

Figura 4.16: Exemplo de configuração do target.properties para robustez

### Gerar monitor a partir da LPS - Planejamento, execução e análise

É necessário a execução de um pequeno roteiro para determinar como será gerado e aplicado um monitor obtido da LPS da FlexMonitorWS. Os passos devem ser seguidos, recomenda-se a aplicação das etapas do método PLUS para gerar um dado monitor. Isso facilitará o entendimento da aplicação da LPS, bem como, a manutenção, se for necessária. Os passos são:

- Análise de Perfis da monitoração

- Configuração da Arquitetura de Linha de Produto
- Configuração do arquivo *properties*
- Execução do produto que contemple a configuração realizada
- Análise e entendimento do resultado gerado pelo Monitor

No item 1, a análise visa entender as necessidades do cliente, estabelecendo para cada necessidade de monitoração um perfil identificado. As perguntas descritas na Seção 2.3 do Capítulo 2 que estabelece o entendimento sobre o problema em questão ajuda a identificar perfis, como por exemplo, quais os alvos da monitoração, quais atributos serão necessários, qual a frequência de execução e tipos de notificação. Para facilitar, utilize a taxonomia descrita na Seção 2.2.1 do Capítulo 2.

Do pressuposto de que no passo 1 as informações foram obtidas, pode ser necessária a instância de uma ou mais ALPs em atendimento aos perfis identificados. Para a instância de uma ALP a configuração do monitor é realizada mediante a seleção das componentes presentes na LPS de acordo com a Figura 4.17.

Como pode ser visto na Figura 4.17, após a criação da instância ela ficará salva em uma configuração de um produto gerado pela LPS (repare Figura 4.17 o arquivo MonitorWS.config).

Após gerar e salvar esta configuração através da FeatureIDE (descrita na Seção 2.4.5), é necessária a execução das etapas de *build*[1] e exportar para um executável (.jar) criando um produto físico.

O passo 3 usa os métodos ligados a plataforma Java (especificamente da biblioteca *java.util.Properties*), como o uso de arquivos *properties (do inglês propriedades)* [Props13]. Este tipo de arquivo permite separar a configuração em blocos, contendo pares de chave e valor. Cada chave representa um parâmetro com seu respectivo valor definido. É possível carregar de forma eficiente este arquivo durante a execução da aplicação e a aplicação consultá-lo quando necessário [Props13].

Para aplicação, convencionou-se o nome **target.properties**. Este arquivo properties é repleto de parâmetros conforme demonstra a Figura 4.18.

O arquivo target.properties é dividido nas seguintes seções:

- ServiceConfig;
- Network;
- Server;
- AppServer;

[1] *Build - construir, refere-se a etapa de compilação*

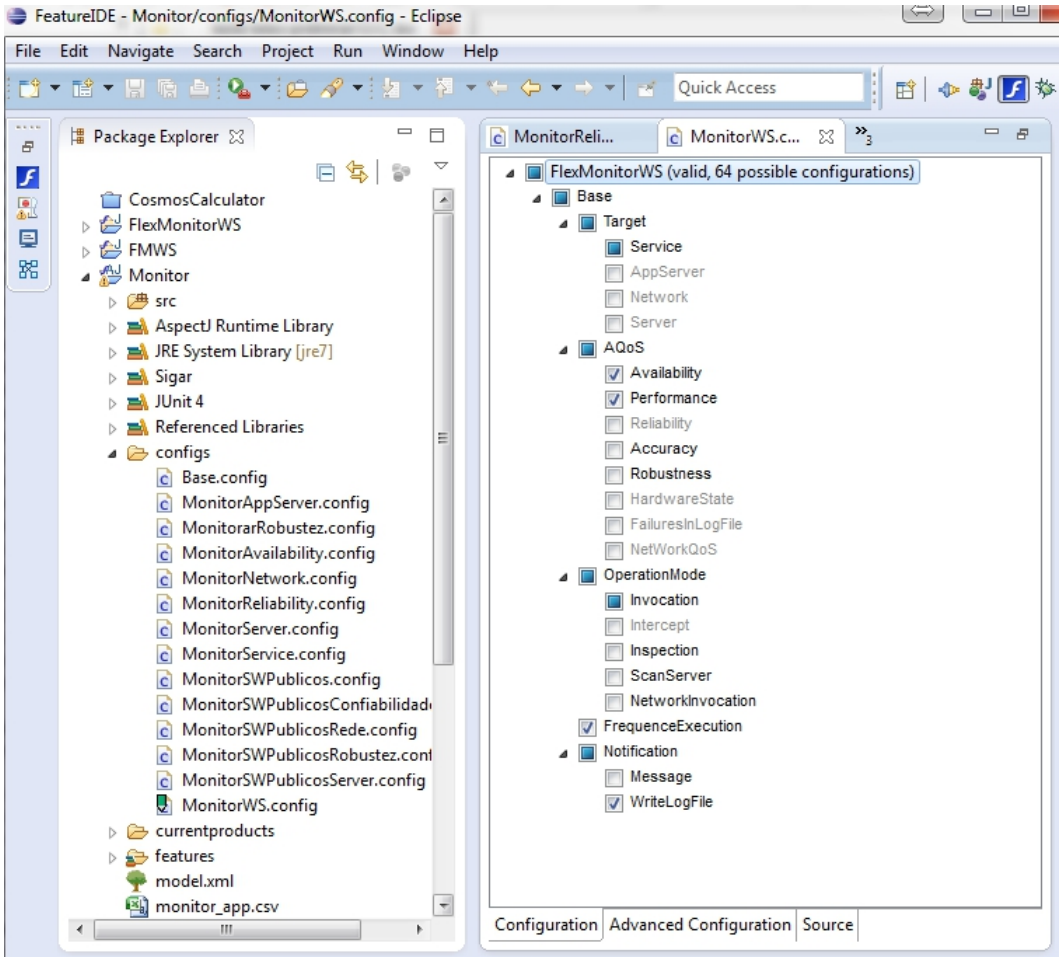



Figura 4.17: Definindo uma configuração para instância de uma ALP

target.properties

```
# Target.properties - Parameter file - FlexMonitorWS
[ServiceConfig1]
service1.serviceName=WSCalc
#Avoid name of endpoint like localhost
#OperationMode in Intercept way not recognize ip like localhost
service1.endpoint=http://192.168.1.13:1244/WSCalc?wsdl
service1.qname=http://test.ws.app/
service1.port=WSCalc
service1.operationName=add
service1.internet = 0
service1.log = 0
service1.active = 0
service1.param1.paramName = valuea
service1.param1.paramValue = 1
service1.param1.paramInvalid = null
service1.param2.paramName = valueb
service1.param2.paramValue = 7
service1.param2.paramInvalid = -1
service1.paramResponseValid = 8
service1.paramResponseInvalid = 0

[Network1]
network1.name = Main Router
network1.ipaddress = 192.168.1.1

[Network2]
network2.name = Servidor ZEUS
network2.ipaddress = 192.168.1.13

[Server1]
server1.nameServer = ZEUS
```

Figura 4.18: *Target.properties* arquivo de parâmetros da FlexMonitorWS

- Notification;
- FrequenceExecution.

Na configuração do Serviço, é inserido parâmetros de conexão, parâmetros de operação e valores de parâmetros das operações. É possível determinar se a configuração ficará ativa durante a monitoração ou não, além de permitir habilitar log da criação do XML de requisição em tempo de execução. Esta informação é necessária para depurar falhas durante a invocação ao serviço. Uma mensagem de falha pode ser retornada do provedor do serviço caso a mensagem de requisição esteja formatada errada.

Podem existir dentro do arquivo N configurações para monitorar N serviços e N recursos. Para tanto, é necessário sempre inserir a identificação do bloco, repetindo as informações dos parâmetros e chaves.

### Execução de um produto

Após a execução dos passos anteriores, é necessário executar o produto gerado pela LPS. Esta execução pode ser realizado diretamente no ambiente de concepção da solução, no caso FeatureIDE ou via execução no console. A execução visa detectar anomalias durante a geração de produtos da LPS, este processo interfere nas regras de composição entre os componentes do produto e comumente alguns erros podem aparecer. Erros de parametrização do arquivo target.properties também são identificados nesta etapa.

### Análise de resultados

Ao executar a monitoração os seguintes arquivos são gerados por recurso:

1. Serviços → monitor-qos.csv
2. Rede → monitor-net.csv
3. Servidor → monitor-server.csv
4. Aplicação Servidora → monitor-appserver.csv

São arquivos textos que contêm o resultado obtido da monitoração de forma tabulado, separado em ponto-virgula (;). Este tipo de arquivo visa facilitar a importação para outros meios como planilhas eletrônicas e a integração com outros sistemas. Durante a execução da monitoração, nem todos os arquivos poderão ser gerados. Isso decorre do fato, de que uma dada configuração da LPS, pode não contemplar um monitor que gere resultados para Serviços, Rede, Servidor ou Aplicação Servidora.

## 4.4 Trabalhos relacionados

De acordo com a Revisão Sistemática da literatura descrita no Capítulo 3, foram traçados alguns desafios e oportunidades e que aparecem como comparativo deste projeto em análise aos estudos primários selecionados durante a revisão.

A proposta descrita por Muller et. al. (2012), SALMonADA efetua uma monitoração para entender contratos SLA, e é baseada na análise relacionada a aplicação da política *WS-Agreement* e a validade dos contratos. A monitoração não ocorre sobre serviços, mas sobre os contratos, com isso é possível monitorar mais de um atributo de QoS, pois cada contrato pode ter um ou mais atributos de QoS.

Em Wetzstein et. Al. (2009), a proposta é orientada para detectar e analisar possíveis fatores que influenciam o desempenho do processo de negócio em duas vertentes: o monitoramento de métricas de PPM (*Process Performance Metrics*) com base na execução das funcionalidades internas de um serviço web, e métricas de atributos QoS. Para realização de métricas de PPM é utilizado um arquivo XML onde são definidos os parâmetros, enquanto que, para atributos QoS existe um monitor que analisa informações temporais (tempo de resposta e disponibilidade) por meio da invocação do serviço.

Em Artaiam et. al., (2008) a proposta é baseada na coleta de dados da aplicação servidora dos serviços Web que estão implantados dentro desta. Algumas aplicações servidoras como Tomcat ou Glassfish de fato emitem estatísticas e podem ser concluídas em quantificar valores de QoS, porém, é restrita aos serviços que estão implantados dentro desta.

Em Souza et. al., (2011) há uma preocupação em determinar o grau invasivo da solução de monitoração no ambiente de modo geral. Este grau invasivo é o quão a solução pode interferir em atributos de QoS como desempenho e disponibilidade, uma vez que se a monitoração ocorrer de forma continua, acarreta no baixo desempenho do serviço e pode ter impactos no serviço ao ponto de este se tornar indisponível.

Um dos principais apelos da proposta Wang et. al., (2009) é não endereçar a monitoração para obter informações restritas a eventos temporais. Para contornar esta restrição, a análise de estado e condição do ambiente passa a ser um importante fator que permite apurar maiores detalhes sobre a degradação de atributos de QoS. Há dois modos de operação fixos, interceptar e instrumentar, além disso, a monitoração passa a funcionar de forma passiva acarretando em entregar valores desatualizados caso nenhuma invocação ao serviço ocorra.

Em Moser et. Al., (2009) a abordagem do VieDAME é composta por três principais funções, monitorar um conjunto de atributos QoS (tempo de resposta, disponibilidade e acurácia), além de aplicar uma adaptação dinâmica na transformação de mensagens compatíveis sintaticamente ou semanticamente de um serviço (XSLT) e na seleção de serviços que foram identificados como substituíveis. Estes dados são armazenados pelo componente que monitora. Se um serviço fora marcado como substituível, o seletor procura

um serviço alternativo, se é encontrado repassa para o componente de transformação de mensagem.

Em Haiteng et. Al., (2011), a proposta tem uma definição de processo onde instâncias de processos BPEL são criadas e somente acionadas quando ocorre a invocação para um determinado serviço. Na invocação, são descobertos serviços com suas respectivas URLs que possam ser conectados. Durante esta etapa, são injetadas as regras de monitoração ou a própria monitoração em si ocorre, utilizando de estratégias de programação orientada a aspectos, em uma camada de interceptação. Com isso, os atributos mencionados são avaliados, coletados e inseridos em um banco de dados.

Em Baresi et. Al. (2008) a proposta aborda o uso de sondas via técnicas de Programação Orientada à Aspectos. Existe uma coleta de dados que explora dois tipos de sondas de dados. As que abordam processos para coletar informações internas (AOP *Probe*) e as que coletam informações externamente ao serviço (*External Probes*). Para realizar a primeira técnica, as sondas são conectadas diretamente ao ambiente do processo em execução. Elas param o processo em execução para garantir a coleta do estado interno. As *Externals Probes* são sondas que podem ser colocadas em todo o ambiente para recolher em tempo de execução as informações do contexto de onde o processo é executado. Uma limitação encontrada é que um das sondas (AOP *Internal Probes*) bloqueiam a execução do serviço para coletar informações, além de ter um alto grau invasivo. Este grau invasivo é maior quando mencionam da necessidade em alterar a máquina BPEL (*BPEL Engine*) para se adequar ao *framework* proposto.

Também em outra proposta de Baresi et. al. (2009) a abordagem utiliza dois *frameworks* baseados em linguagens para definir parâmetros de monitoração por meio de WSCoL (*Web Service Constraint Language*) e RTML (*Run-Time Monitor specification Language*). A primeira diz respeito as propriedades internas e externas que serão utilizadas para coletar eventos relevantes e intrínsecos ao que se deseja necessariamente monitorar. Enquanto que RTML está mais interessada em definir parâmetros que irão monitorar as propriedades de composição. A junção das duas abordagens ao definir as linguagens e as fórmulas de resultados, oferece estatísticas sobre cada parâmetro monitorado.

A Tabela 4.3 representa a classificação de algumas vantagens da FlexMonitorWS em comparação aos estudos identificados durante a aplicação da revisão sistemática. Na primeira coluna temos uma relação destas vantagens. O primeiro item é a independência de plataforma de aplicação servidora. A ausência de independência de plataforma implica na baixa portabilidade da proposta de monitoração. Muitas propostas estudadas como representado na tabela executam do lado do servidor vinculado a um Container (e.g. Tomcat), e interceptam dados dentro deste container.

Tabela 4.3: Comparativo dos trabalhos relacionados

| **Vantagens** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **Flex Monitor WS** |
|---|---|---|---|---|---|---|---|---|---|---|
| Independência de plataforma da aplicação servidora | • | • | | | • | | | | • | • |
| Permite definir Perfis de monitoração | • | • | | • | | | | • | • | • |
| Existe nível de flexibilidade com atributos de QoS | • | | | • | • | • | • | • | • | • |
| Número de Atributos de QoS verificados e validados | 2 | 2 | 6 | 3 | 2 | 3 | 3 | 3 | NA | 7 |
| Existe nível de flexibilidade com modos de operação | | | | | | | | • | • | • |
| Monitora mais de um alvo simultaneamente | | | | | • | | | • | | • |
| Utiliza abordagem LPS | | | | | | | | | | • |

**Legenda:** 1) Muller12, 2) Wetzstein09, 3) Artaiam08, 4) Souza11, 5) Wang09, 6) Halima08, 7) Haiteng11, 8) Baresi08, 9) Baresi09
• A proposta considera uma abordagem semelhante a vantagem
**NA** Nenhum atributo de QoS avaliado na abordagem

## 4.5 Resumo do capítulo

Este capítulo abordou toda a solução proposta da aplicação, divididas em quatro fases apoiadas pelo Método PLUS: Modelagem de requisitos, Modelo de Análise OA (Orientado a Aspectos), Projeto de LPS e Implementação, Implantação e execução. Cada fase, após entendida foi aplicada e descrita de forma detalhada. Anteriormente ao emprego da primeira, uma visão geral sobre a FlexMonitorWS foi oferecida de modo a iniciar com alguns artefatos necessários como o documento de visão e o documento de casos de uso.

Ao final do detalhamento da fase final, foi destacado detalhes de implementação e execução de um produto que corresponde a uma configuração de uma instância da ALP, demonstrando desde a geração de um produto da LPS até a execução.

Ainda, o capítulo fez um comparativo das vantagens que conseguimos obter da solução com algumas das soluções vistas na literatura a partir da Revisão Sistemática da Literatura descrita no Capítulo 3.

# Capítulo 5

# Estudos de caso

Este capítulo apresenta três estudos de caso e suas avaliações. Os estudos foram analisados de forma qualitativa e baseados nas diretrizes estabelecidas no guia proposto por Runeson & Host (2009), para condução de estudos de caso em engenharia de software.

Segundo Runeson & Host (2009) a condução de Estudos de Caso deve ter um objetivo claro, uma explanação sobre o caso estudado, uma teoria, questões de pesquisa, métodos e estratégia de solução.

No primeiro estudo de caso, descrito na Seção 5.1, consideramos um cenário controlado, com alguns elementos inerentes ao mundo real. O foco principal era avaliar a viabilidade da solução.

O segundo estudo de caso, descrito na Seção 5.2, a solução foi aplicada em serviços Web públicos considerando um cenário real. A aplicação sobre tais serviços poderiam determinar também a viabilidade da solução em monitorar serviços Web (SOAP), incluindo serviços públicos em um cenário não controlado. Estes serviços podem ser monitorados do lado do consumidor e outros pontos que poderiam degradar atributos de QoS. Os seguintes alvos também foram monitorados, Rede (e.g. IPs gateway ligado ao Servidor), Servidor (do lado do consumidor monitorar memória, disco, tempo de uso).

No terceiro e último estudo de caso, descrito na Seção 5.3, foi aplicada a FlexMonitorWS para injetar falhas e identificar vulnerabilidades em serviços Web. Para este estudo de caso, foi considerado um serviço Web fictício e serviços Web públicos do cenário real. O principal objetivo deste estudo foi determinar a viabilidade da FlexMonitorWS como solução de injeção de falhas.

## 5.1 Estudo de caso 1: Cenário Controlado

Este estudo de caso tem como foco principal a avaliação da flexibilidade da solução na monitoração de diferentes atributos de QoS, diferentes recursos da infraestrutura de TI de diferentes modos de operação. Os serviços Web monitorados são considerados simples e compostos, retratando elementos do mundo real, em um cenário ligado a ambientes SOA.

**Questões de pesquisa aplicada ao estudo**

Em suma, como objeto principal a flexibilidade, este primeiro estudo de caso deveria responder de forma satisfatória às seguintes perguntas:

1. É possível atender a diferentes atributos de QoS na monitoração?
2. É possível atender a diferentes recursos de infraestrutura de TI ligado a serviços Web?
3. É possível cruzar os dados da monitoração e entender onde a causa dos problemas ocorrem?

- $H_1$ Existe flexibilidade para acomodar diferentes atributos de QoS e diferentes alvos de monitoração.
- $H_2$ Há uma eficiência da solução em detectar anomalias em cenários por meio do cruzamento de dados provenientes da monitoração.

### 5.1.1 Recursos usados

Os recursos físicos e virtuais usados para implementação do estudo de caso são representados na Tabela 5.1.

Tabela 5.1: Recursos físicos utilizados para o estudo de caso 1

| **Servidor** | **Especificação** |
|---|---|
| ZEUS | Processador Core 2 Quad 2.13GHz, 64bits, 4 GB Ram, 1 TB, Windows 7 64Bits |
| NERO | Processador Core i7 2.30Ghz, 6 GB Ram, 1 TB HD, Windows 8 64bits |
| VMLINUXMINT | Processador Dual Core, 2 Gb Ram, Linux Mint 16 Cinnamon |
| VMUBUNTU | Processador Dual Core, 2 GB Ram, Ubuntu 12.04 Desktop |
| As máquinas virtuais foram implantadas em uma máquina AMD PhenonX6 1.80Ghz, 4 Gb Ram, para instalação de cada nó virtual foi utilizado o XenServer 6.20 (versão *Trial*). | |

### 5.1.2 Sobre o Cenário Controlado

O cenário controlado é representado pela Figura 5.1 onde os recursos são: um Provedor Master, que efetua uma composição de serviços, consumindo serviços do Provedor A e B. O Consumidor A por sua vez, consome o serviço do Provedor Master.

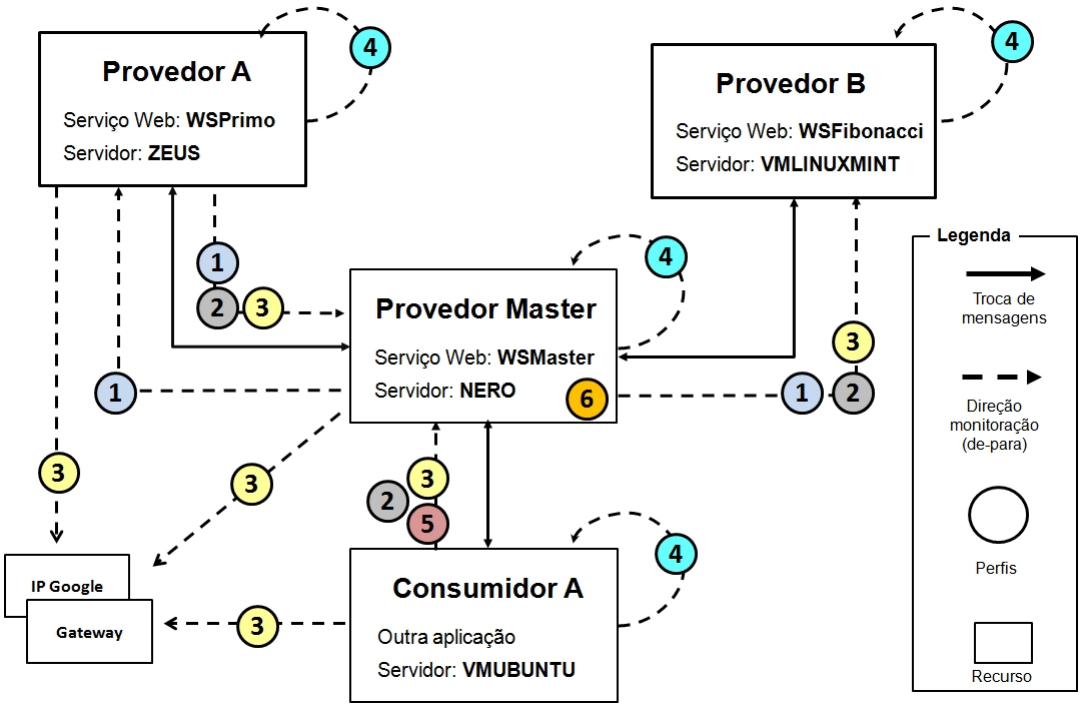



Figura 5.1: Elementos que compõem o cenário controlado

Os perfis de monitoração são representados através dos círculos contendo um número, e é explicada detalhadamente por meio da Tabela 5.2. Cada pequeno círculo com seu respectivo número representa um perfil e tem um papel distinto no cenário, e foi inserido estrategicamente para coletar dados de forma a produzir uma monitoração eficiente.

Os serviços inseridos nos recursos foram implementados especificamente para atendimento ao cenário em questão. São eles: um serviço que calcula se o número é primo (WSPrimo), um serviço que devolve o ultimo fator da sequência de Fibonacci (WSFibonacci), um serviço composto agregando WSPrimo e WSFibonacci, que resultou no serviço composto WSMaster. E por último uma aplicação que consome o WSMaster.

### 5.1.3 Planejamento de execução

De acordo com o cenário, partiu do pressuposto da existência de Acordos em nível de Serviço (SLA), sendo:

- Provedor Master estabeleceu que a disponibilidade do Serviço provido pelo Provedor A, deveria ser de 98% por hora, inferior a esse acordo o Provedor A estaria sujeito a penalidades

- Consumidor A estabeleceu que o atributo Confiabilidade (*Reliability*) deveria ser acrescentado ao fornecimento do Serviço provido pelo Provedor Master

- Consumidor A estabeleceu que o atributo de desempenho do Serviço provido pelo Provedor Master não deveria ser maior que 200ms em média por hora por requisição.

Para todos os perfis a Notificação seria por arquivo de Log e a Frequência de Execução seria de 30 segundos. Para cada perfil de monitoração, foi instanciada a arquitetura da Linha de Produto para gerar cada monitor que representasse o Perfil elencado na Tabela 5.2. Ao final da identificação dos perfis, gera-se uma família de produtos baseada em LPS (e.g. monitorarDisponibilidade.jar, monitorarDesempenho.jar, etc).

Tabela 5.2: Perfis de monitoração identificados no cenário

| N. | De (Recurso) | Para (Alvo) | Atributo de QoS | Modo de Operação |
|---|---|---|---|---|
| 1 | Prov. Master | Prov.A e Prov B | Disponibilidade | Invocação |
| 1 | Prov. A | Prov. Master | Disponibilidade | Invocação |
| 2 | Prov. Master | Prov. B | Desempenho | Invocação |
| 2 | Prov A | Prov. Master | Desempenho | Invocação |
| 2 | Cons. A | Prov. Master | Desempenho | Invocação |
| 3 | Prov. Master | Prov. B, Gateway e IP Google | Disponibilidade e Desempenho da rede | Invocação |
| 3 | Prov. A | Prov. B, Gateway e IP Google | Disponibilidade e Desempenho da rede | Invocação |
| 3 | Cons. A | Prov. B, Gateway e IP Google | Disponibilidade e Desempenho da rede | Invocação |
| 4 | Servidores | Todos monitoram a si próprio | Estado/cond. servidor | Inspeção |
| 5 | Cons. A | Prov. Master | Confiabilidade | Intercept. |
| 6 | Prov. Master | Container Glassfish | Falhas em Log | Inspeção |

Para gerar os perfis de monitoração, quatro pastas foram criadas, uma para cada ponto de monitoração e ali inseridos os executáveis e a configuração (target.properties) necessária para atender a cada perfil desejado.

### 5.1.4 Execução e expectativas

Todos os produtos foram copiados para uma área compartilhada na rede em comum a todas as máquinas do cenário. Após, foi executada a monitoração tendo um tempo inicial de corte previsto, isso devido ao fato de não ter sido iniciada a monitoração simultaneamente em cada ponto.

Durante a execução que compreendeu em um intervalo de 14 horas, a análise se deu através de uma sequência de interrupções previstas gerando falhas no ambiente para

Tabela 5.3: Interrupções previstas para avaliar comportamentos

| N. | Descrição da Interrupção | Hora prev. | Duração | Objetivos |
|---|---|---|---|---|
| 1a. | Interrupção interface de rede do Provedor Master com o consumo ocorrendo no Consumidor A. | 1h | 20 min. | Avaliar a solução, comportamento do serviço e impacto sobre atributo disponibilidade. |
| 2a. | Interrupção serviço em execução no Provedor A com o consumo ocorrendo no Provedor Master e no Consumidor A | 8:30h | 1 hora | |
| 3a. | Gerar uma carga na banda de rede do Provedor Master com o consumo ocorrendo no Consumidor A | 9:30h | 2 horas | Avaliar o comportamento da solução de monitoração e do impacto sobre os atributos de QoS de disponibilidade e desempenho. |
| 4a. | Remoção da Política WS-RM (*Reliable Messaging*) do Provedor Master | 13h | 10 minutos | Avaliar o comportamento da solução de monitoração ao detectar a ausência da entrega confiável de mensagem (ligada ao atributo QoS confiabilidade) |

avaliar no cenário, o comportamento da solução de monitoração, o comportamento do serviço Web e o impacto sobre os atributos de QoS monitorados.

A ideia principal, era gerar um conjunto de interrupções em intervalos de tempo configurando um cenário atípico, porém factível. Após gerada cada interrupção, aguardar

um tempo para posterior análise e retomar o cenário para um patamar de normalidade, onde todos os elementos funcionam normalmente para aplicar a próxima interrupção elencada na Tabela 5.3.

### 5.1.5 Resultados

A partir da aplicação dos diversos monitores no cenário e por meio da geração de interrupções elencadas na Tabela 5.3, apresentamos agora os resultados obtidos.

#### Primeira e segunda interrupções no cenário

Ambas interrupções (primeira e segunda) visam entender a partir dos resultados e do cruzamento dos dados resultantes da monitoração, o atributo de QoS disponibilidade e onde está a causa da interrupção.

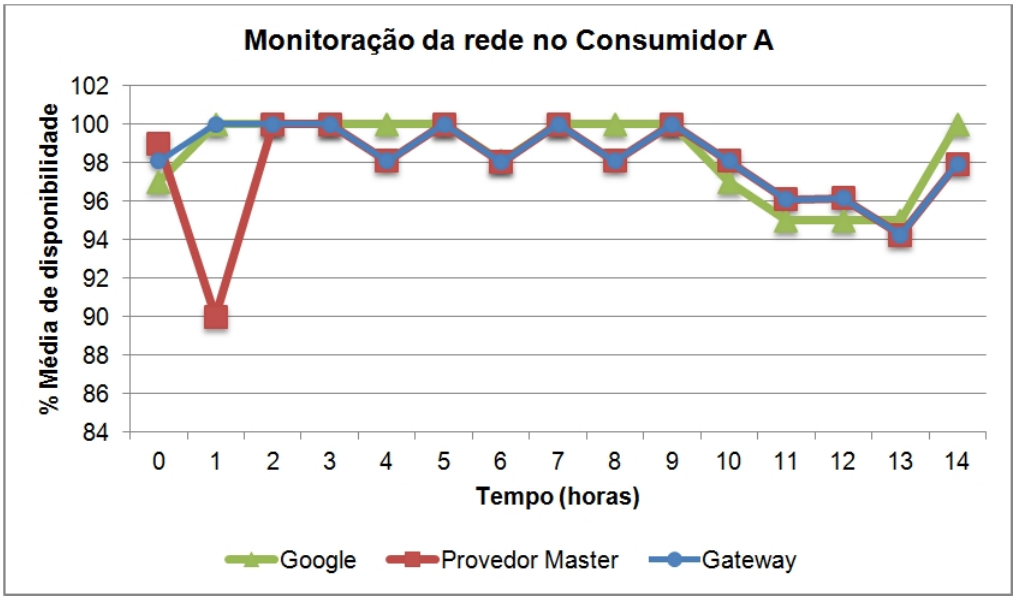



Figura 5.2: Gráfico demonstrando monitoração da rede a partir do Consumidor A em alguns alvos estratégicos

A Figura 5.2 representa um cruzamento de dados obtidos da monitoração do IP que representa o Provedor Master, *Gateway* e um IP público, neste caso o do Google. Note na Figura 5.2, durante a primeira hora de monitoração, foi executada a primeira interrupção. Repare na Figura 5.2, que o percentual médio da disponibilidade por hora cai e retoma durante a partir da segunda hora de monitoração.

A Figura 5.3 representa o atributo de QoS disponibilidade obtida durante a monitoração do Provedor Master para os alvos, Provedor A e Provedor B. Como a primeira interrupção ocorreu na desativação da interface de rede durante a primeira hora no Provedor Master, é possível perceber esta falha no gráfico. Por meio deste mesmo gráfico

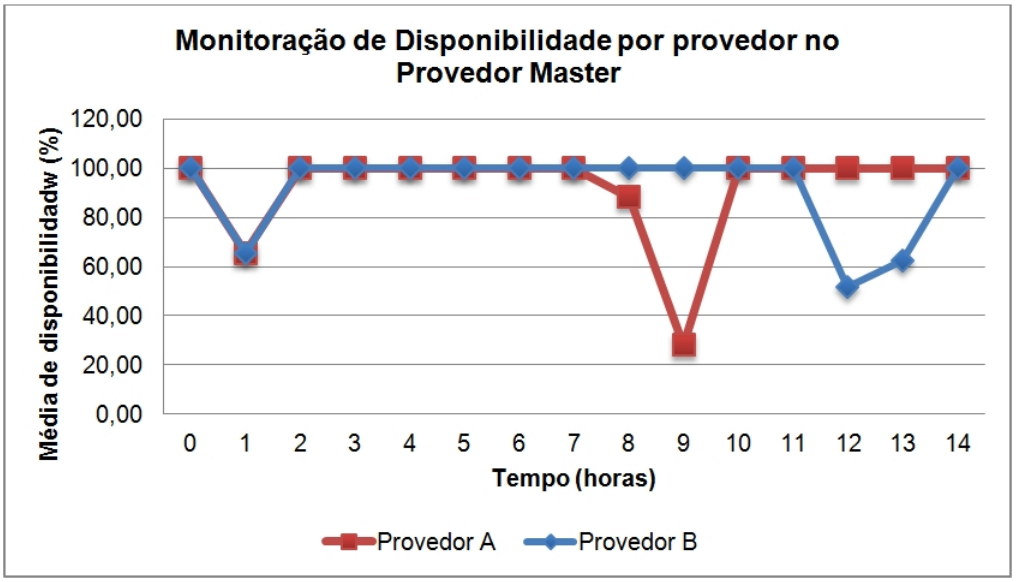



Figura 5.3: Monitoração do atributo de QoS de disponibilidade no Provedor Master monitorando outros provedores

(Figura 5.3), durante a oitava e nona hora, analisando o serviço do Provedor A, o percentual médio da disponibilidade cai, isso significa que houve uma indisponibilidade neste intervalo e que retomou a partir da décima hora (10h). Esta última análise está ligada a segunda interrupção.

Ainda nesta mesma análise, porém, analisando agora a Figura 5.4 tendo a monitoração do lado do Consumidor A, é possível concluir que não houve falha durante a nona hora (9h) na infraestrutura do Provedor Master. Era possível acessar o Provedor B, contudo, embora não representado no gráfico da Figura 5.4, houve um impacto no Consumidor A, pois quando a falha ocorreu no Provedor A se propagou para o Provedor Master ocorrendo a indisponibilidade do serviço no Master também, assim, afetando o Consumidor A.

### Cruzamento de dados para detectar a causa onde a interrupção ocorreu

Se efetuar a seguinte questão analisando os gráficos das figuras acima – A primeira interrupção, durante a primeira hora (1h) ocorreu de fato em quais dos pontos do cenário? Pressupondo o não conhecimento das interrupções.

Através dos gráficos representados pelas Figuras 5.2, 5.3 e 5.4 é possível determinar que a falha poderia estar entre o Provedor Master e o Provedor A, pois ambos apresentaram o mesmo declínio do atributo de QoS disponibilidade durante a primeira hora.

No caso apresentado na Figura 5.5, mesmo que o Provedor Master não soubesse da indisponibilidade proveniente de uma interrupção na primeira hora, a monitoração do Provedor A sobre o Provedor Master e sobre o IP Público do Google iria garantir a precisão do diagnóstico oferecido pela solução de monitoração. Assim, mesmo que o

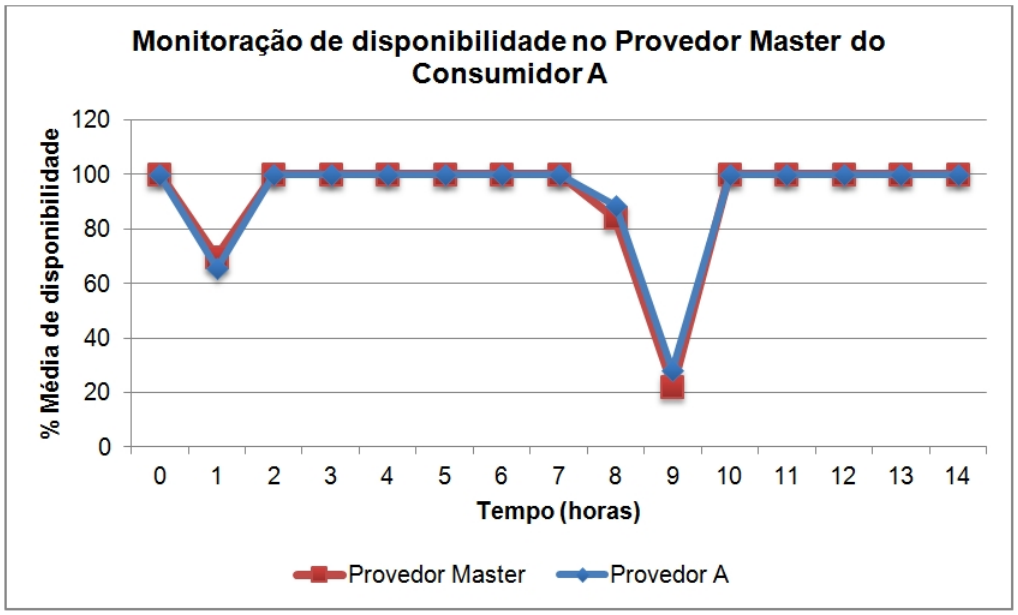



Figura 5.4: Gráfico demonstrando o cruzamento dos dados da monitoração no Provedor Master e no Consumidor A

Provedor Master queira eventualmente se isentar de culpa da degradação, o Provedor A pode apontar o real causador da degradação, para este caso. Os resultados obtidos foram do monitor de rede apontado para IP Público Google (Perfil 3 da Figura 5.1) e monitor de disponibilidade do serviço do Provedor Master (Perfil 2 da Figura 5.1).

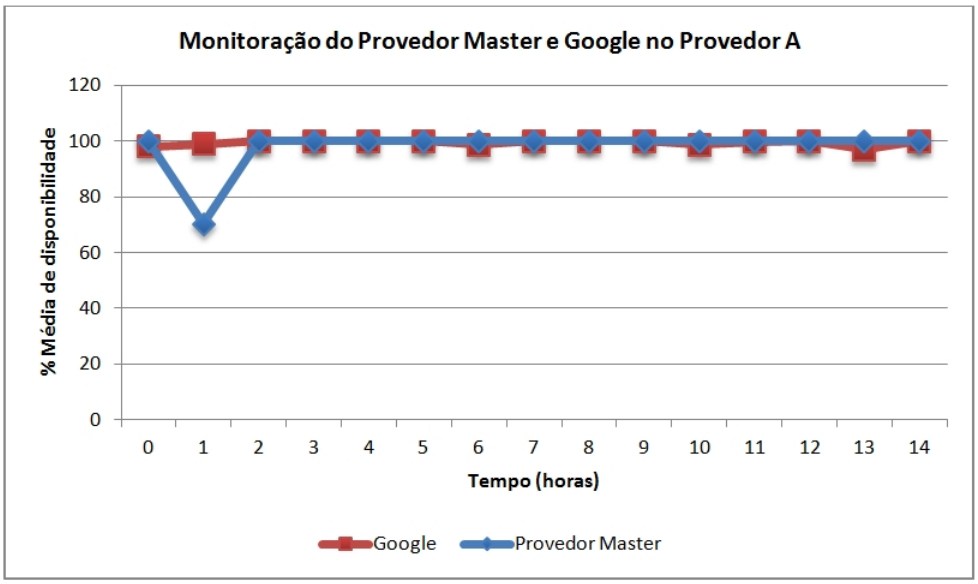



Figura 5.5: Monitoração do Provedor Master e Google diretamente do Provedor A

Para inferir esta situação, é necessária a existência de uma entidade intermediária que

monitora os pontos em casos que exista um SLA. Um exemplo prático seria, o próprio provedor monitora seu consumidor com quem tem um contrato firmado (SLA). Aqui neste caso específico, inserimos um Monitor no Provedor A para colher dados da disponibilidade do seu consumidor, no caso o Provedor Master. Um monitor no serviço Web do Provedor Master e um monitor de rede tendo como alvo o IP público como o Google. Através do cruzamento de dados destes dois monitores seria possível avaliar quem é o responsável pela indisponibilidade do serviço e onde ocorreu exatamente a causa da interrupção.

### Terceira interrupção

A terceira interrupção é gerada uma alta carga da banda de rede do Provedor Master, de tal maneira, a criar oscilações no atributo de QoS de desempenho.

A Figura 5.6, apresenta os dados de desempenho capturados através da monitoração executada do Provedor A e do Consumidor A, para a rede e o serviço Web no Provedor Master, durante a terceira interrupção.

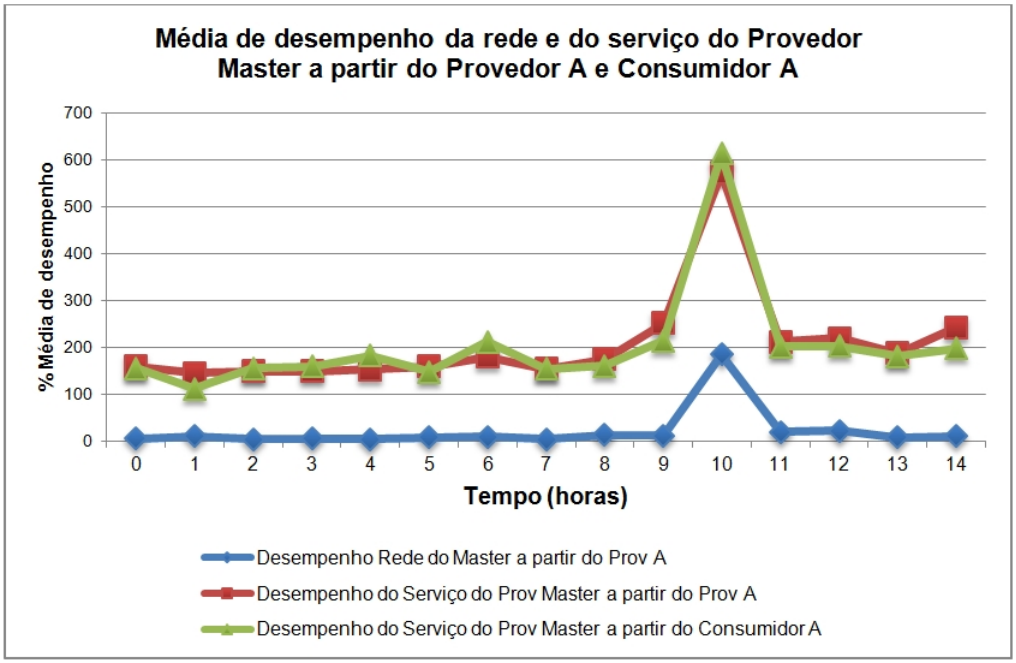



Figura 5.6: Gráfico representando monitoração de desempenho do Provedor Master a partir de outros pontos de monitoração

Note na Figura 5.6, entre as horas 9h e 11h, ocorre um pico e o atributo de QoS de desempenho sofre uma degradação no Provedor Master.

Para o desempenho, a degradação é perceptível e tem impactos diretos no Consumidor A. Isso está em conformidade com o previsto para a terceira interrupção, dado que o serviço Web provido por Master é afetado pela carga de dados na rede ocasionando o baixo desempenho. Este efeito é propagado no consumidor A e também pode ser detectado por

uma monitoração estratégica no Provedor A. Mantivemos no Provedor A um monitor apontando para rede do Provedor Master. É possível avaliar esta situação no gráfico representado na Figura 5.6.

Perceba no gráfico da Figura 5.6, quanto maior o tempo de resposta médio para executar uma dada operação, menor é o desempenho do serviço Web.

### Quarta interrupção

A quarta interrupção, é a remoção da Política WS-RM (*ReliableMessaging*) do Provedor Master. Esta interrupção visa entender sobre a monitoração do atributo de QoS de confiabilidade (do inglês *reliability*).

O serviço Web executava habilitado com a entrega confiável (WS-RM) e era possível detectar através da monitoração por interceptação executada no Provedor Master.

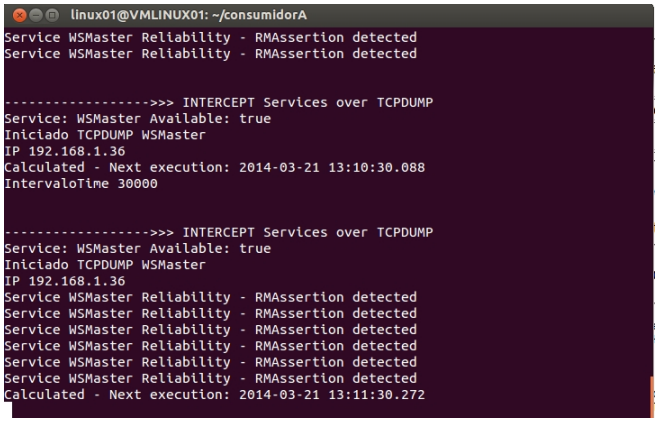



Figura 5.7: Monitoração detecta mensagem com WS-RM implementado para o Serviço Web no Provedor Master

Note na Figura 5.7, algumas das interceptações que ocorreram durante a monitoração para o serviço Web (do Provedor Master) é possível identificar a existência da política WS-RM (*Reliable Messaging*).

Posteriormente, para intervir no cenário, é removida a política do serviço Web no Provedor Master e então a monitoração passa a detectar esta anomalia. Como pode ser visto nos dizeres da Figura 5.8, "RMAssertion NOT detected". Isso era previsto, e significa que não há a implementação da política que fora descrito na Seção 2.3.3. Portanto, o serviço não contém o atributo de QoS confiabilidade.

### Violações do acordo SLA

Durante a monitoração, um dos objetivos principais é fiscalizar se um dado atributo de QoS que está em um SLA, respeitou os limites previamente estabelecidos.

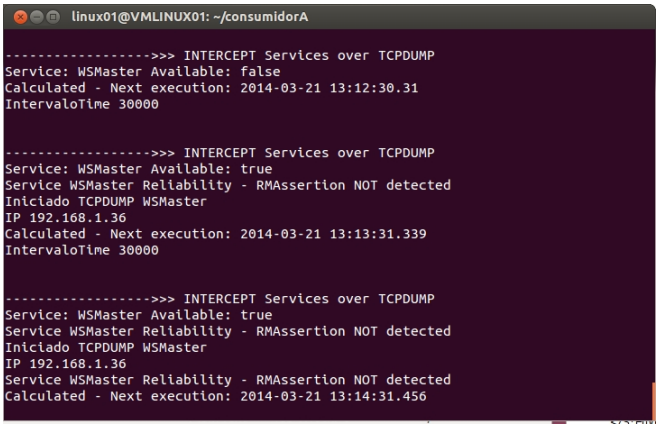



Figura 5.8: Monitoração detecta mensagem com WS-RM implementado para o Serviço Web no Provedor Master

Tabela 5.4: Relatório de violação com as medições realizadas

| Acordo com | Atributo QoS | Limites SLA | Medição Contratante | Medição Contratado |
|---|---|---|---|---|
| Provedor Master com Provedor A | Disp. | Acima da média de 98%/hora | Média de 92,11%/hora | Média de 98,02%/hora |
| Consumidor A com Provedor Master | Desemp. | Média de 200 ms/hora | 207,46 ms/hora | 203,54 ms/hora |

Note na Tabela 5.4, no acordo existente entre Master com Provedor A, o atributo de QoS disponibilidade pela medição realizada pelo Provedor Master o percentual médio por hora do valor do atributo fica abaixo do limite acordado em SLA. Porém, com a medição realizada também pelo Provedor A, a média fica exatamente no limite estabelecido em acordo, não configurando a violação do acordo SLA. Neste caso, onde há uma divergência da média obtida em ambos os pontos de comunicação, o Provedor A tem os dados colhidos sob a rede do Provedor Master comprovando a falha ocorrida na rede durante a primeira interrupção e isso corrobora com a degradação do atributo de QoS disponibilidade do lado do Provedor Master, mas sem que a média fosse afetada no Provedor A.

No caso do atributo de QoS de desempenho, na medição realizada pelo Consumidor A (contratante) o valor fica abaixo do limite estabelecido em acordo com o Provedor Master (contratado), configurando um desempenho abaixo do satisfatório.

### 5.1.6 Avaliações sobre o estudo de caso 1 - Cenário controlado

A partir dos resultados gerados da execução do estudo de caso, é possível responder as questões endereçadas para este estudo de caso:

1. **É possível atender a diferentes atributos de QoS na monitoração?**

   Durante a execução, foi possível avaliar e verificar a monitoração de disponibilidade de serviços Web, disponibilidade da rede, desempenho de serviços Web e desempenho da rede e confiabilidade, embora, outros atributos de QoS pudessem ser gerados pela LPS. A monitoração dos demais atributos de QoS, está via ilustrações e gráficos na Seção Apêndices.

2. **É possível atender a diferentes recursos de infraestrutura de TI ligado a serviços Web?**

   Durante a execução, a monitoração ocorreu sobre serviços Web, rede, servidor e aplicação servidora.

3. **É possível cruzar os dados da monitoração e entender onde a causa dos problemas ocorrem?**

   Os dados da monitoração de serviços Web foram cruzados com dados da monitoração da rede, este cruzamento permitiu entender melhor onde uma dada causa ocorreu, veja exemplo em Resultados na Seção 5.1.5 – Primeira e segunda interrupções.

## 5.2 Estudo de Caso 2: Cenário Real

Neste estudo de caso, optamos em realizar uma monitoração em serviços Web públicos a fim de testar a viabilidade da FlexMonitorWS em monitorar diferentes serviço Web, avaliando diferentes atributos de QoS em um cenário real.

A monitoração, neste caso, tem um perfil do lado do consumidor, a avaliação foi feita em recursos físicos como monitorar o servidor, rede e o serviços Web.

### Questões de pesquisa aplicada ao estudo

Para avaliar sob a ótica da viabilidade da solução, o que também está ligado a nossa hipótese do projeto, elaboramos as seguintes questões de pesquisa para este estudo de caso:

1. É possível detectar anomalias em serviços Web públicos usando a solução proposta?

2. O estudo pode ser replicado para diferentes cenários?

3. Qual a precisão do diagnóstico emitido pela monitoração realizada pela FlexMonitorWS?

- $H_1$ A FlexMonitor é viável e pode ser replicada para cenários não controlados.

A primeira questão, está ligada a eficiência em detectar anomalias durante a monitoração destes serviços. A segunda questão está ligada a capacidade da solução ser replicada para outros cenários diferentes do aplicado no primeiro estudo de caso. Seu comportamento durante a execução, evidencia a viabilidade da solução, uma vez que se ela monitora em diferentes cenários, logo a solução pode ser tornar viável. A terceira questão será avaliada sob a comparação dos dados da monitoração da rede e dos serviços Web em cruzamento com os diferentes pontos de monitoração.

### 5.2.1 Recursos usados

Foi utilizado um único computador com capacidade de monitorar alguns serviços Web elencados para a monitoração. Este computador é o ZEUS contendo Processador Core 2 Quad 2.13GHz, 64bits, 4 GB Ram, 1 TB, Windows 7 64Bits.

### 5.2.2 Sobre o Cenário Real

O cenário real é composto por quatro serviços Web públicos. Três deles são serviços equivalentes onde tem uma operação em comum de validação de número de cartão de crédito. Um outro serviço efetua uma validação de e-mail.

Para entender possíveis desajustes na degradação dos atributos de QoS elencados para o cenário, foi inserido um quinto elemento onde a monitoração foi feita em um IP público sendo nesse caso o IP do Google.

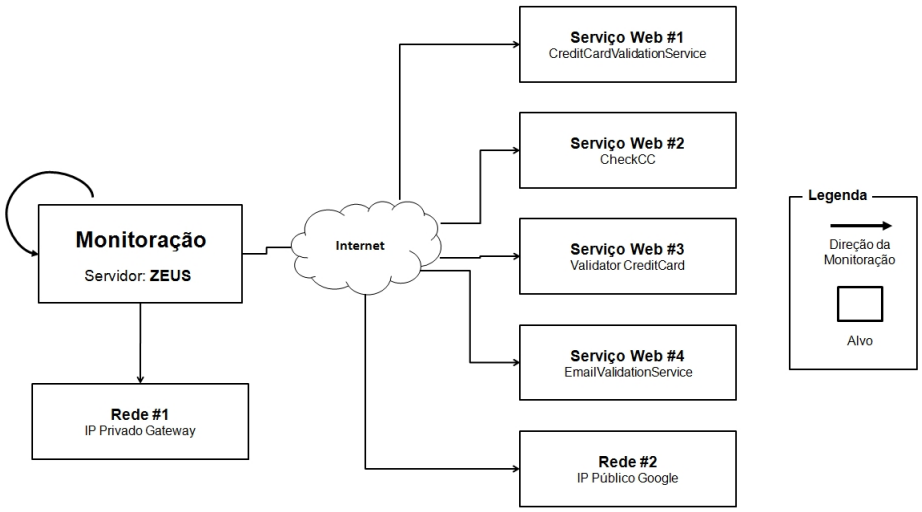



Figura 5.9: Elementos que compõem o cenário real

Conforme Figura 5.9, no elemento central (servidor: ZEUS), a monitoração ocorre sobre os recursos físicos, memória, disco e interface de rede. Um elemento adjacente dentro da rede permitiu entender em casos onde se houvesse falha na rede interna do

elemento central seria detectada pela monitoração. O alvo de monitoração neste caso é o *Gateway* (Rede 1 – Figura 5.9).

Perceba na Figura 5.9, a monitoração sobre o *Gateway* está mais ligada a questão três sobre a precisão do diagnóstico da solução. Se a rede falhar, a monitoração deveria apontar anomalia para todos os serviços, se a rede não falhar e a avaliação detectar anomalias dos serviços Web de forma individual, também indica uma resposta positiva a esta questão.

### 5.2.3 Planejamento de execução

Para efetuar a monitoração do lado do consumidor e atender ao cenário em questão, foi necessário gerar uma família de produtos da LPS, e criar quatro monitores para atender aos perfis de monitoração elencados na Tabela 5.5.

Tabela 5.5: Perfis de monitoração identificados para o cenário

| **Perfil de monitoração** | Alvo | **Atributos de QoS** | **Modo** de **operação** |
|---|---|---|---|
| Monitor AQoS | Todos serviços | Disponibilidade, desempenho, acurácia e robustez | Invocação |
| Monitor WS-RM | Todos serviços | Confiabilidade a partir da política WS-RM (*Reliable Messaging*) | Interceptação |
| Monitorar Rede | Gateway e IP Público (Google) | Disponibilidade e desempenho | Invocação |
| Monitorar Zeus | ZEUS | Estado e condição do hardware | Inspeção |

Todos os perfis de monitoração tendo uma frequência de execução a cada minuto, e a notificação por meio de arquivo de Log que seria usada posteriormente para análise e resultado.

### 5.2.4 Execução

Após o início da monitoração, foi realizada uma prévia checagem dos primeiros dados a respeito de cada alvo a fim de garantir um bom funcionamento e de uma boa qualidade dos dados. Estas primeiras coletas são analisadas averiguando eventuais falhas e erros de

configuração e parametrização realizados no arquivo target.properties. Se houvesse alguma falha, o monitor era parado, ajustado parâmetros no arquivo e reiniciado a monitoração.

A execução compreendeu em 17 dias de monitoração. A análise foi feita através do cruzamento dos diversos dados provenientes da monitoração.

### 5.2.5 Resultados

Para análise dos dados, os arquivos gerados foram exportados para planilhas eletrônicas e gerados os gráficos para análise. Em um primeiro momento, consideramos apenas cálculos básicos de estatísticas das médias aritméticas simples dos valores para cada atributo.

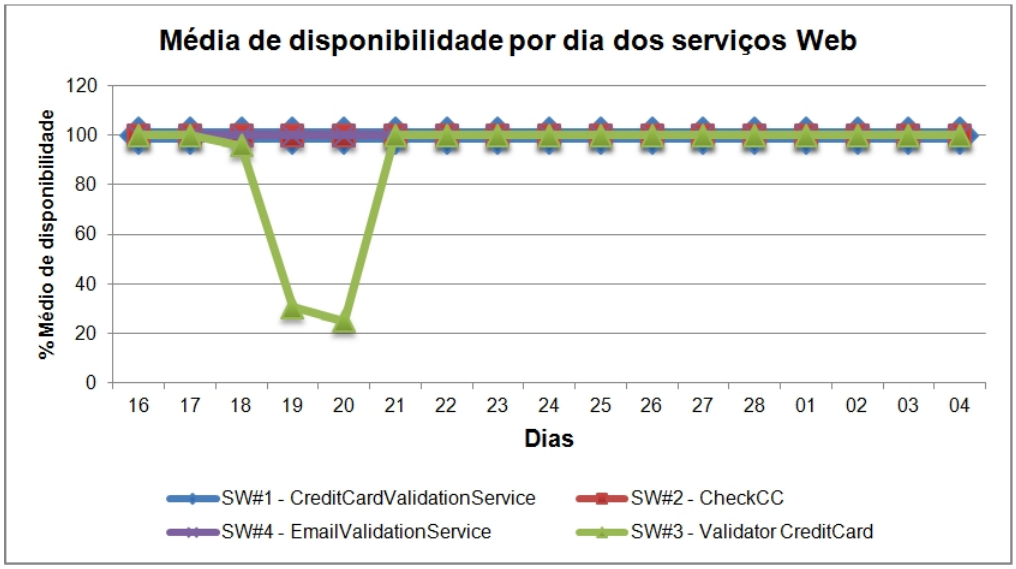



Figura 5.10: Gráfico da média de disponibilidade dos serviços Web por dia dos serviços Web monitorados

Conforme é apresentado na Figura 5.10, é perceptível que o Serviço Web 3 que efetua a validação de um número de cartão de crédito, teve seu pico de degradação da sua disponibilidade no dia 19 e 20, se restabelecendo a partir do dia 21 e entrando em normalidade.

Uma análise por hora ajuda no entendimento da degradação do atributo de QoS disponibilidade ao oferecer uma média por hora de todos os dias monitorados. Para atender a esta necessidade e efetuar uma análise mais apurada, a Figura 5.11 apresenta o percentual da média da disponibilidade por hora.

Note na Figura 5.11, há um cruzamento de dados da monitoração dos serviços Web elencados e dos pontos de rede como IP do Google e o IP do Gateway (rede interna do ZEUS). Com este cruzamento é possível avaliar que não houve falha na rede durante o cálculo da média, uma vez que no gráfico, para Gateway e Google a média da disponibilidade se mantém bem próxima da média dos serviços Web em normalidade. Contudo, o

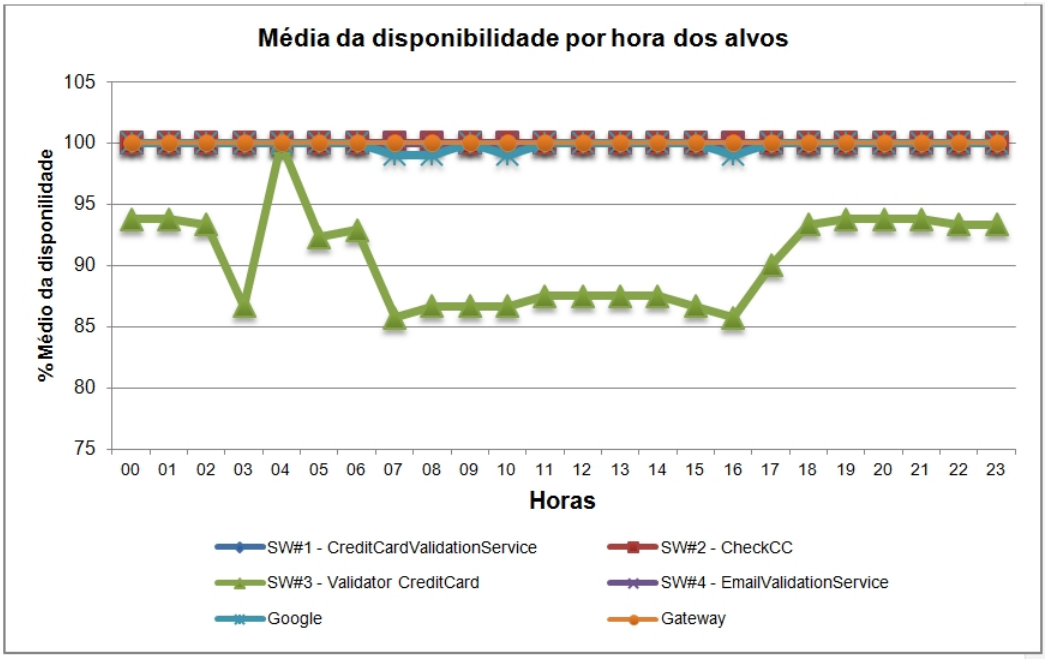



Figura 5.11: Gráfico da média de disponibilidade calculada por hora dos serviços Web, Google e Gateway

Serviço Web 3 - *Validator CreditCard* tem uma degradação razoável a comparar com os demais alvos apresentados no gráfico.

### Atributo de QoS: Desempenho

Na Figura 5.12, temos a média de desempenho por hora para alvos, como serviços Web, bem como os recursos da rede como IP público do Google e o IP privado do Gateway. Através do cruzamento dos dados da monitoração da rede e dos serviços Web, é possível detectar algumas anomalias.

A primeira, com relação ao serviço Web 2 - *CheckCC* (validador de cartão de crédito), durante a hora treze (13h) é possível detectar uma anomalia. O atributo de QoS de desempenho atinge um pico acima de 1000 milissegundos (ms) na média por hora. Isso é decorrente da degradação no próprio provedor deste serviço. No gráfico, é possível constatar que na rede (analisando Gateway) não houve interferência, bem como, nos demais serviços Web a média dos valores obtidos é mantida.

A segunda anomalia é em relação ao Serviço Web 3, onde o desempenho é bem abaixo da média a comparar com os demais serviços Web. Porém, o desempenho mesmo com degradação a comparar com os demais, é bem próximo do constante, tendo oscilações para mais ou para menos de 600ms na média por hora, não ultrapassando 800ms, nem se aproximando a 400ms.

A partir das duas anomalias apontadas no gráfico, é possível concluir que, o baixo

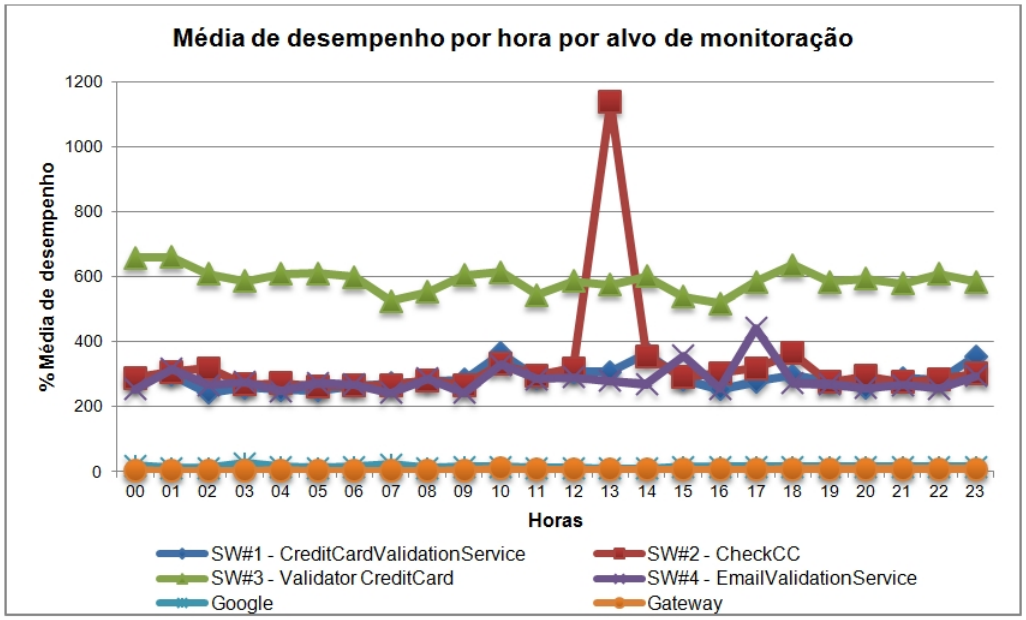



Figura 5.12: Média de desempenho por hora por alvo de monitoração

desempenho está no próprio provedor do serviço de ambos os provedores. Porém, há diferenças nas duas situações: a primeira está no serviço Web 3 onde há uma média com pequenas oscilações. Enquanto, a segunda situação o serviço Web 2 a monitoração por hora detectou um pico na hora 13 de forma diária.

Para o caso da hora 13 para o serviço Web 2 (SW2-*CheckCC*), uma análise mais ampla é necessária. Até para estabelecer um juízo com mais acurácia sobre o comportamento do serviço em questão. A Figura 5.13 apresenta um gráfico comparando, o desvio padrão, a média, a mediana e a moda por hora dos valores obtidos durante a monitoração.

É perceptível no gráfico da Figura 5.13, que exatamente o valor máximo da hora treze (13h) extrapola o desempenho a comparar com outros valores estatísticos. E mesmo comparando com os valores máximos para as outras horas, o valor é ponderado e próximo da média. A evidência clara obtida, é que foi uma hora atípica do uso comum do serviço. Contudo, se os valores utilizados for parte de contratos em nível de serviço (SLA), e for obtido a partir da média, o atributo de QoS teria seu valor degradado em função de uma única hora que o provedor não foi capaz de atender ao desempenho estabelecido. Mas, mesmo assim é uma análise que depende do consenso de ambos os lados, provedor e consumidor.

Na Tabela 5.6 é possível obter mais valores estatísticos que apoiam a análise. Na Tabela 5.6, a média, o desvio padrão, a mediana e a moda apresentam valores ponderados em milissegundos. O desvio padrão é um pouco mais que o dobro da média, isso pode representar que os valores estão espalhados, ou seja, há uma dispersão.

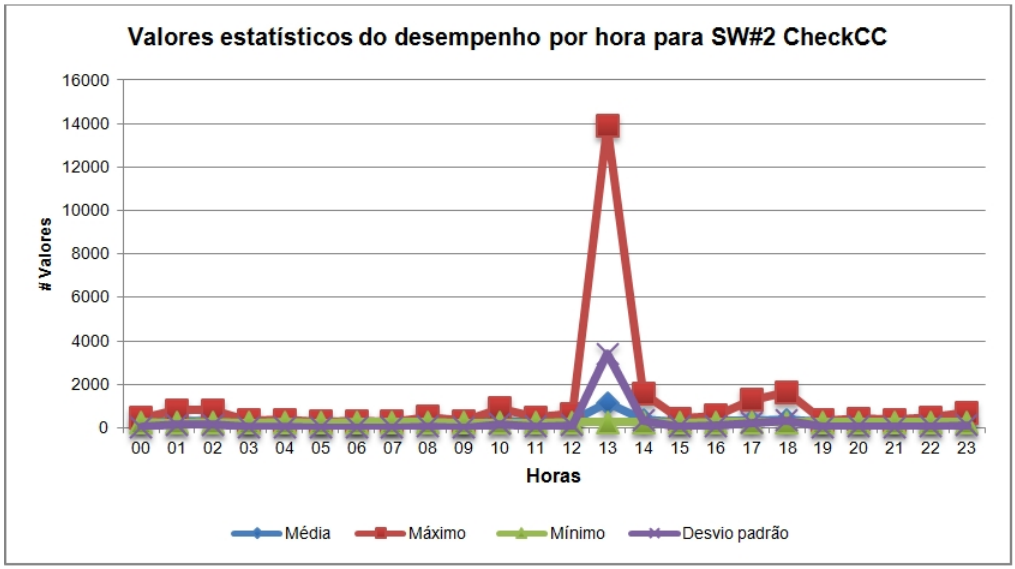



Figura 5.13: Gráfico para análise estatística de SW2-*CheckCC*

Tabela 5.6: Valores estatísticas sobre o SW2

| Termo estatístico | Valor (ms) |
|---|---|
| Média | 333,84 |
| Desvio padrão | 719,30 |
| Mediana | 266 |
| Moda | 250 |

**Confiabilidade**

A partir da Figura 5.14, o resultado da monitoração dos serviços Web sobre o atributo de QoS confiabilidade é apresentado. Repare que a maioria dos serviços elencados para o estudo não implementam a política WS-RM (WS-*ReliableMessaging*).

O serviço Web 2 não permite a interceptação de mensagens. Este serviço utiliza criptografia baseada no protocolo HTTPS. Como dito na Seção 4.3.4 (Detalhes da implementação), a solução de monitoração por interceptação usa o TCPDUMP e não obtém dados criptografados.

Na Tabela 5.7 tem-se a relação de serviços Web e outros recursos e os valores médios diários obtidos da monitoração. Os valores apresentados são para os atributos de QoS analisados durante o estudo de caso 2 - Cenário real.

### 5.2.6 Avaliações sobre o estudo de caso 2 - Cenário real

A partir da execução do estudo de caso 2, é possível responder as questões elencadas para o estudo:

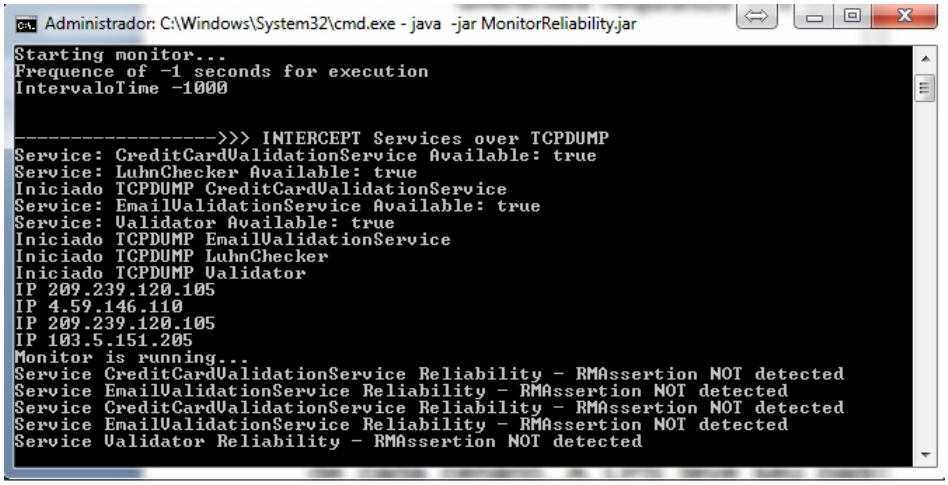



Figura 5.14: Monitoração do atributo de QoS confiabilidade dos serviços Web elencados

Tabela 5.7: Média diária dos atributos de QoS

| Serviço | Disponibilidade | Desempenho | WS-RM |
|---|---|---|---|
| SW1 | 100% | 287 ms | Não |
| SW2 | 100% | 331ms | Não |
| SW3 | 91,25% | 590ms | Não |
| SW4 | 100% | 283ms | Não |
| Google | 97,9 | 14ms | NA |
| Gateway | 100% | 5ms | NA |

1. **É possível detectar anomalias em serviços Web públicos usando a solução proposta?**

   Durante a análise dos resultados é possível evidenciar de forma satisfatória e positiva que a solução tem capacidade de detectar anomalias em serviços Web públicos.

2. **O estudo pode ser replicado para diferentes cenários?**

   Ao executar este estudo de caso e o anterior é possível evidenciar de forma satisfatória que a solução pode ser replicada para diferentes cenários com serviços Web, recursos de infraestrutura de TI e de diferentes modos de operação. Ambos cenários dos estudos de caso aplicados tem elementos bem diferentes de um para o outro.

3. **Qual a precisão do diagnóstico emitido pela monitoração realizada pela FlexMonitorWS**

Sobre o resultado obtido durante a monitoração ligado a questão 3, temos duas importantes considerações: a primeira é sobre 1) a validade dos dados obtidos sobre a análise de diferentes protocolos de rede, 2) a análise dos dados via cruzamento de dados sem considerar protocolo, e sim o resultado geral para o atributo de disponibilidade.

Consideremos que, 1) é necessária a comparação com outras soluções na literatura que tenha monitorado um cenário semelhante, o que não seria trivial. Há uma outra maneira de saber da validade dos dados ao comparar os dados da monitoração via invocação para o atributo de disponibilidade do serviço Web com a monitoração via rede (*ping*) do IP do *endpoint* do serviço. Porém, os dados de ambas monitorações contêm distorções, uma vez que, utilizamos diferentes protocolos durante a monitoração. Para monitorar serviços Web durante a invocação foi usado o protocolo TCP que contém garantias de entrega. Para a rede, a monitoração ocorreu via comando ping usando o protocolo ICMP, qualquer ocorrência de *timeout*, é calculado como indisponível, afetando a média de disponibilidade do serviço.

A partir da consideração 2) sobre a análise dos dados do atributo de QoS de disponibilidade, independente da forma como fora obtido, é possível inferir algo. Na Figura 5.10 a monitoração de serviços Web avaliando disponibilidade foi capaz de detectar anomalias identificadas em alguns serviços Web (e.g. SW2 e SW3), anomalias estas que foram apresentadas de forma individual. Pelo cruzamento dos dados da monitoração da rede e dos serviços Web é possível concluir com mais acurácia estas anomalias analisando a Figura 5.11. Em uma eventual falha da rede, a monitoração detectaria esta falha também e apresentaria a degradação da disponibilidade em todos os alvos, incluindo serviços Web.

A partir desta análise é possível evidenciar uma resposta positiva e satisfatória para a questão três.

Os dados resultantes da monitoração deste estudo podem ser utilizados para eventuais substituições necessárias de serviços Web com operações equivalentes, casos como os serviços Web com operação de checagem de número de cartão de crédito foco principal deste estudo de caso. Estes dados resultantes da monitoração podem ser facilmente aplicados em abordagens como adaptação dinâmica e recuperação.

## 5.3 Estudo de caso 3: Cenário de Injeção de Falhas

Neste estudo de caso, aplicamos a FlexMonitorWS para identificar vulnerabilidades em serviços Web em cenário controlado e em todos os serviços públicos do cenário real, determinando a robustez para cada serviço Web.

O estudo de caso irá garantir a FlexMonitorWS como uma solução para monitoração, como também, contribui com a aplicação de técnicas de injeção de falhas.

As injeções são aplicadas baseadas em scripts de substituição, discutidas na Seção 2.3.5 e será utilizada o processo de injeção de falhas discutido em detalhes de implementação na Seção 4.3.4.

**Questões de pesquisa aplicada ao estudo**

Para avaliar sob a ótica da viabilidade da solução como injetora de falhas, elaboramos as seguintes questões de pesquisa para este estudo de caso:

1. É possível detectar vulnerabilidades em serviços Web através da solução?

2. Existe flexibilidade em executar os scripts de substituição definidos pelo usuário em ataques?

3. Qual a precisão do diagnóstico emitido pela FlexMonitorWS ao identificar vulnerabilidades em serviços Web?

- $H_1$ A FlexMonitor é viável também para injetar falhas do tipo substituição para uma gama de tipos de ataques por scripts definidos de forma automática e manual.

A primeira questão visa concluir se de fato vulnerabilidades são detectadas pela FlexMonitorWS e quais tipos de ataques surtiram resultados satisfatórios, promovendo uma discussão sobre comportamentos de serviços para cada vulnerabilidade identificada. A segunda, se a primeira contém resposta positiva e satisfatória, existe flexibilidade da solução em executar os scripts manuais definidos pelo usuário da solução. A terceira questão visa determinar o quão válido é o diagnóstico apontado pela FlexMonitorWS.

Está última questão é ligada a uma análise mais apurada dos dados, e que é papel da solução juntamente a um técnico ou administrador do serviço. A solução apresenta resultados, contudo, a montagem do script manual não é de responsabilidade da solução. Além disso, falsos positivos podem aparecer em soluções do tipo escaners de vulnerabilidades e soluções de injeção de falhas. Cabe ressaltar, que é fundamental uma boa interpretação dos dados resultantes da solução.

### 5.3.1 Recursos usados

Foi utilizado um único computador com capacidade para execução dos objetivos planejados neste estudo de caso. Este computador é o ZEUS contendo Processador Core 2 Quad 2.13GHz, 64bits, 4 GB Ram, 1 TB, Windows 7 64Bits.

### 5.3.2 Sobre o Cenário de Injeção de Falhas

O cenário é composto por quatro serviços Web públicos e um serviço Web criado especificamente para este estudo de caso, mas que é semelhante aos serviços do cenário controlado. Este serviço foi denominado de serviço interno. Os serviços web públicos são os mesmos aplicados no estudo de caso 2. O serviço Web interno é executado em ambiente local a FlexMonitorWS.

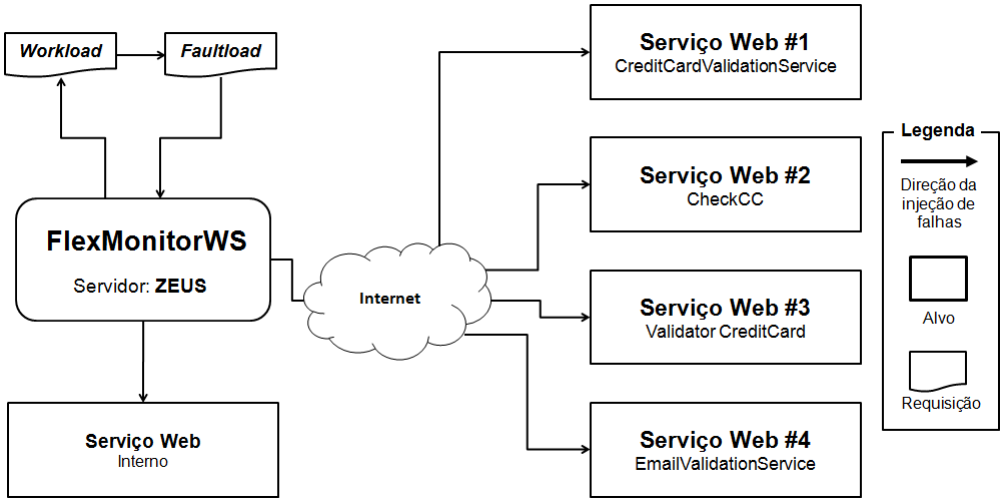



Figura 5.15: Elementos que compõem o cenário de injeção de falhas

Na Figura 5.15, temos a FlexMonitorWS implantada no servidor ZEUS e é o ponto escolhido para disparar a injeção de falhas. O serviço interno à rede, cujo objetivo é esclarecer melhor sobre a execução de um mesmo script, para todos os demais serviços. O que contribui em responder a terceira questão aplicada ao estudo. Ao comparar a execução com o comportamento do serviço interno, com os demais serviços em relação a um mesmo tipo de ataque.

### 5.3.3 Planejamento e Execução

Note na Figura 5.15, a carga de trabalho (*workload*) será montada utilizando a primeira requisição ao serviço utilizada para verificar a disponibilidade. Após, esta carga de trabalho é modificada inserindo os scripts configurados no target.properties ou gerados automaticamente para o disparo contra os serviços Web. A modificação da carga de trabalho, como dito na Seção 2.3.5 (Robustez), cria uma carga de falhas (*faultload*). Este processo ocorre em tempo de execução. Cada script gera uma nova carga de falha.

Para a execução do estudo de caso, foi gerado apenas um produto MonitorRobustness.jar como é apresentado na Figura 5.16. Após a geração, foi realizado a configuração do arquivo target.properties, que determina a forma de execução dos scripts, o conjunto de scripts manuais, alvos, meios de notificação e da frequência de execução do monitor. Os scripts configurados automaticamente são apresentados na Tabela 5.8.

Scripts manuais também foram considerados durante a execução para dois serviços, o serviço interno e o serviço Web 1 (CredicardValidationService). Os scripts foram, um *Malformed XML* contendo um padrão semelhante ao primeiro script, relacionado na Tabela 5.8 e um script do tipo *XML Bomb*.

Para o script *Malformed XML 3* da Tabela 5.8, o valor válido inserido é comparado

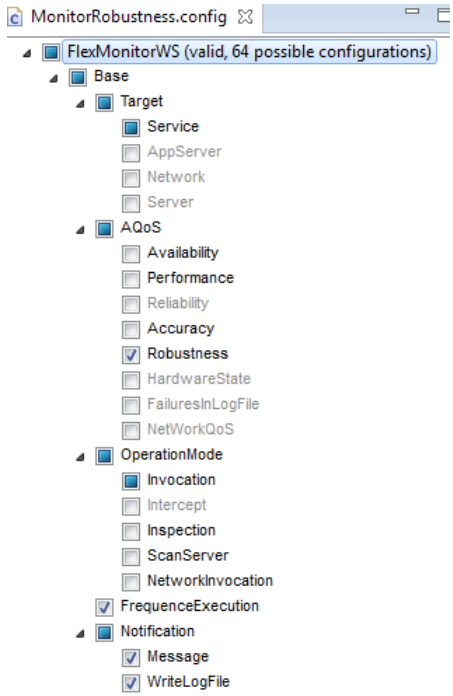



Figura 5.16: Configuração do produto MonitorRobustness.jar

Tabela 5.8: Scripts para criação da *faultload*

| Nome script | Descrição | Status Esperado |
|---|---|---|
| *Malformed XML 1* | Não fechar tag de parâmetros da requisição | 400 ou 500 |
| *Malformed XML 2* | Alterar tags padrão da mensagem SOAP | 400 ou 500 |
| *Malformed XML 3* | Inserir um valor juntamente com um valor válido | 400 ou 500 |
| *Malformed XML 4* | Inserir novos atributos (e.g. repetir 10 vezes a cadeia <teste></test> | 400 ou 500 |
| *XML Bomb* | Enviar antes do inicio do corpo da mensagem um script de expansão contendo código malicioso | 400 ou 500 |
| *Duplicate Request* | Envio de requisições em duplicidade | 400 ou 500 |

com uma resposta esperada válida do serviço. Por exemplo, para verificar se um número de cartão de crédito é válido, informamos um número de cartão realmente válido e sabemos que o retorno será verdadeiro. Mas, ao executar este script, o valor deverá ser falso, uma vez que juntamente com o número do cartão é enviado outros valores ou caracteres

especiais (e.g. %$@), correspondente a um tipo de ataque. Se o serviço retornar um valor válido, é uma vulnerabilidade de alto risco.

### 5.3.4 Resultados

Para análise dos resultados, a solução emite um relatório para cada serviço demonstrando a execução de cada script.

#### Serviço Web Interno

Por meio do log visualizado do lado do servidor, a execução de cada script para o serviço Web interno, pode ser conferido e analisado o comportamento ao ataque, uma vez que o serviço opera local junto com a FlexMonitorWS.

O relatório representado na Figura 5.17, mostra o resultado para as três execuções, incluindo execução de scripts automática e manual. Note na Figura 5.17, a última coluna do relatório determina se foi identificada uma vulnerabilidade (VF) ou não (VNF).

```
--------------------------------------------------------------------------------
Robustness report - Service: WSCalc
URL: http://127.0.0.1/WSCalc?wsdl
Script Name                          Code Expected          Code Returned   Vulnerability

Malformed XML 1 - Tags from WS       400 500                500             VNF - Passed
Malformed XML 2 - Standard tags      400 500                500             VNF - Passed
Malformed XML 3 - Special param      400 500                200              VF - FAILED
Malformed XML 4 - New parameter      400 500                200              VF - FAILED
XML Bomb injection                   400 500                500             VNF - Passed
Duplicate XML Requests               400 500                500             VNF - Passed
Manual 1 - Invalid Type              400,500                200              VF - FAILED
Manual 2 - MalFormed XML             400,500                500             VNF - Passed
Manual 3 - Bomb Injection            400,500                500             VNF - Passed
Params/Values: valueb - Input: A
Malformed XML 3 - Special param - Response expected: 0 - Response returned: 0
WSCalc
        - Robustness calculated: 3.0
```

Figura 5.17: Relatório de execução dos scripts e identificação de vulnerabilidades para o serviço interno

Os dois primeiros scripts apontados no relatório tem retorno duvidoso (código de status HTTP 500), uma vez que, uma resposta robusta seria o retorno do código de status HTTP 400. Contudo, nenhuma divulgação de informações confidenciais do serviço foram expostas no retorno.

A primeira vulnerabilidade encontrada no relatório representado na Figura 5.17, é relativa ao script *Malformed XML 3 - Special param* e é um falso positivo, uma vez que, mesmo a inserção de um parâmetro inválido, este por sua vez, foi convertido para nulo ou zero do lado do servidor. Isso pode ser conferido diretamente na penúltima linha do

relatório, onde é descrito os valores de resposta esperado e retornado. A comparação de ambos conclui o falso positivo para este específico caso.

A segunda vulnerabilidade de fato, é uma vulnerabilidade identificada pela solução do lado do servidor. Embora, nenhuma exceção foi gerada à execução e é tratada como uma requisição normal. A Figura 5.18, o XML de requisição (*faultload*) e o XML de resposta obtida do servidor após processar o serviço interno. O servidor pode neste caso ter simplesmente ignorado. Porém, este tipo de *faultload* pode representar a possibilidade de enviar um grande número de novos parâmetros e gerar uma sobrecarga no servidor e este tornar-se indisponível. Portanto, consideremos com uma falha injetada com sucesso culminando em uma vulnerabilidade identificada do serviço interno.

```
<soap:Envelope
    xmlns:soap="...">
    <SOAP-ENV:Header />
    <SOAP-ENV:Body>
        <ns2:add xmlns:ns2="http://test.
        ws.app/">
            <test></test>
            <test></test>
            <test></test>
            <test></test>
            (..repeat for N times..)
            <test></test>
            <valuea>12</valuea>
            <valueb>73</valueb>
        </ns2:add>
    </SOAP-ENV:Body>
</soap:Envelope>
```

```
<?xml version="1.0" ?>
<S:Envelope
    xmlns:S="http://schemas.
    xmlsoap.org/soap/envelope/">
    <S:Body>
        <ns2:addResponse
        xmlns:ns2="http://test.ws.app/">
            <return>85</return>
        </ns2:addResponse>
    </S:Body>
</S:Envelope>
```

Figura 5.18: XML de requisição/resposta representando Vulnerabilidade Encontrada

A terceira vulnerabilidade é a respeito do script *Invalid Type* penetrado no servidor de forma manual. Contudo, a injeção de falha não foi caracterizada como tal e a resposta é normal. Do lado do servidor nenhuma exceção é gerada e o valor convertido para nulo ou zero. Este script é semelhante ao script *Malformed XML 3 - Special param*, e que também não obteve sucesso no ataque.

Além das vulnerabilidades identificadas, tentamos executar o ataque do tipo *XML Bomb injection* atráves de uma execução automática e uma manual. A única diferença de uma execução para outra, está no local onde foi inserido o script, tentando forçar a expansão de um dado elemento, porém, sem sucesso. A Figura 5.19 apresenta o script de uma das execuções.

### Serviço Web 1 - CreditCardValidationService

A Figura 5.20 apresenta o relatório de execução dos scripts como resultado aos ataques aplicados ao SW 1. As vulnerabilidades encontradas são semelhantes as apresentadas para o serviço interno. Contudo, há uma diferença importante, o código de status de retorno é

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE xmlbomb [
    <!ELEMENT data (#PCDATA)>
    <!ENTITY a '&b;&b;&b;'>
    <!ENTITY b 'foo'>]>
    <data>&a;</data>
<soap:Envelope xmlns:soap="..."
    xmlns:SOAP-ENV="..."
    xmlns:xsd="..."
    xmlns:xsi="...">
    <SOAP-ENV:Header />
    <SOAP-ENV:Body>
        <ns2:add xmlns:ns2="http://test.ws.app/">
            <valuea>&a;</valuea>
            <valueb>73</valueb>
        </ns2:add>
    </SOAP-ENV:Body>
</soap:Envelope>
```

Figura 5.19: *XML Bomb injection* utilizado nos scripts

mais robusto em comparação com o serviço interno, o código 400 informa ao requisitante do serviço que ele está efetuando uma *Bad request*, em face a *faultload* disparada ao servidor.

A única vulnerabilidade encontrada de fato é a relacionada ao script *Malformed XML 4 - New parameter*, como dito anteriormente para o serviço interno, esta falha permite gerar ataques que torna o serviço indisponível ou DoS (do inglês *Denial of Service*).

```
--------------------------------------------------------------------------------
Robustness report - Service: CreditCardValidationService
URL: http://www.ezzylearning.com/services/CreditCardValidationService.asmx?wsdl
Script Name                          Code Expected          Code Returned   Vulnerability

Malformed XML 1 - Tags from WS       400 500                400             VNF - Passed
Malformed XML 2 - Standard tags      400 500                400             VNF - Passed
Malformed XML 3 - Special param      400 500                200              VF - FAILED
Malformed XML 4 - New parameter      400 500                200              VF - FAILED
XML Bomb injection                   400 500                400             VNF - Passed
Duplicate XML Requests               400 500                400             VNF - Passed
Manual 1 - Malformed XML             400,500                400             VNF - Passed
Manual 2 - Bomb Injection            400,500                400             VNF - Passed
Manual 2 - Invalid Types             400,500                400             VNF - Passed
Params/Values: cardNumber - Input: 4111111111111111%$@
Malformed XML 3 - Special param - Response expected: false - Response returned: false
CreditCardValidationService
        - Robustness calculated: 2.0
```

Figura 5.20: Relatório de execução dos scripts e identificação de vulnerabilidades para o SW 1

### Serviço Web 2 - CheckCC

A execução do conjunto de scripts automáticos para este serviço, culminou em identificar não uma vulnerabilidade, mas sim uma falha de alto risco. A primeira vulnerabilidade

identificada é sobre a inserção de uma cadeia de caracteres após um valor válido, de acordo com o relatório apontado na Figura 5.21. Para este caso específico, o número de cartão de crédito inserido é válido, porém, ao acrescentar-se caracteres especiais ao final do valor, ele se torna inválido, criando-se uma requisição maliciosa correspondente a uma injeção de falha. Observe na Figura 5.21, valores destacados apontam o diagnóstico oferecido pela solução.

```
--------------------------------------------------------------------------------
Robustness report - Service: LuhnChecker
URL: https://ws.cdyne.com/creditcardverify/luhnchecker.asmx?WSDL
Script Name                          Code Expected          Code Returned   Vulnerability

Malformed XML 1 - Tags from WS       400 500                400             VNF - Passed
Malformed XML 2 - Standard tags      400 500                400             VNF - Passed
Malformed XML 3 - Special param      400 500                200              VF - FAILED
Malformed XML 4 - New parameter      400 500                200              VF - FAILED
XML Bomb injection                   400 500                400             VNF - Passed
Duplicate XML Requests               400 500                400             VNF - Passed
Params/Values: CardNumber - Input: 4111111111111111%$@
Malformed XML 3 - Special param - Response expected: false - Response returned: true
LuhnChecker
        - Robustness calculated: 3.0
```

Figura 5.21: Relatório de execução dos scripts e identificação de vulnerabilidades para o SW 2

Para assegurar a veracidade do resultado obtido pela FlexMonitorWS, executamos o mesmo script em duas soluções que permitem executar testes aos serviços por meio de invocações normais e inserimos este valor de entrada como inserido durante o script (valor do número de cartão de crédito: 4111111111111111%$@). A Figura 5.22 apresenta o resultado em uma das soluções testadas, como no caso a solução de testes SoapUI.

#### Resultado final sobre a robustez dos serviços Web

Para os demais serviços, os resultados são semelhantes as vulnerabilidades detectadas identificadas no serviço interno e no SW 1. Contudo, o SW 3, estava indisponível durante a realização deste estudo de caso e não foi possível aferir qualquer resultado sobre as injeções de falhas.

A Tabela 5.9 apresenta este resultado e confirma, para o conjunto de scripts executados foi possível assegurar que nenhum serviço deste estudo de caso possui robustez. Inclusive, para o serviço que efetua uso de protocolo para transmissão de comunicação segura (HTTPS) como o caso do SW 2.

### 5.3.5 Avaliações sobre o estudo de caso 3

A partir da execução do estudo de caso 3 é possível responder as questões elencadas para o estudo:

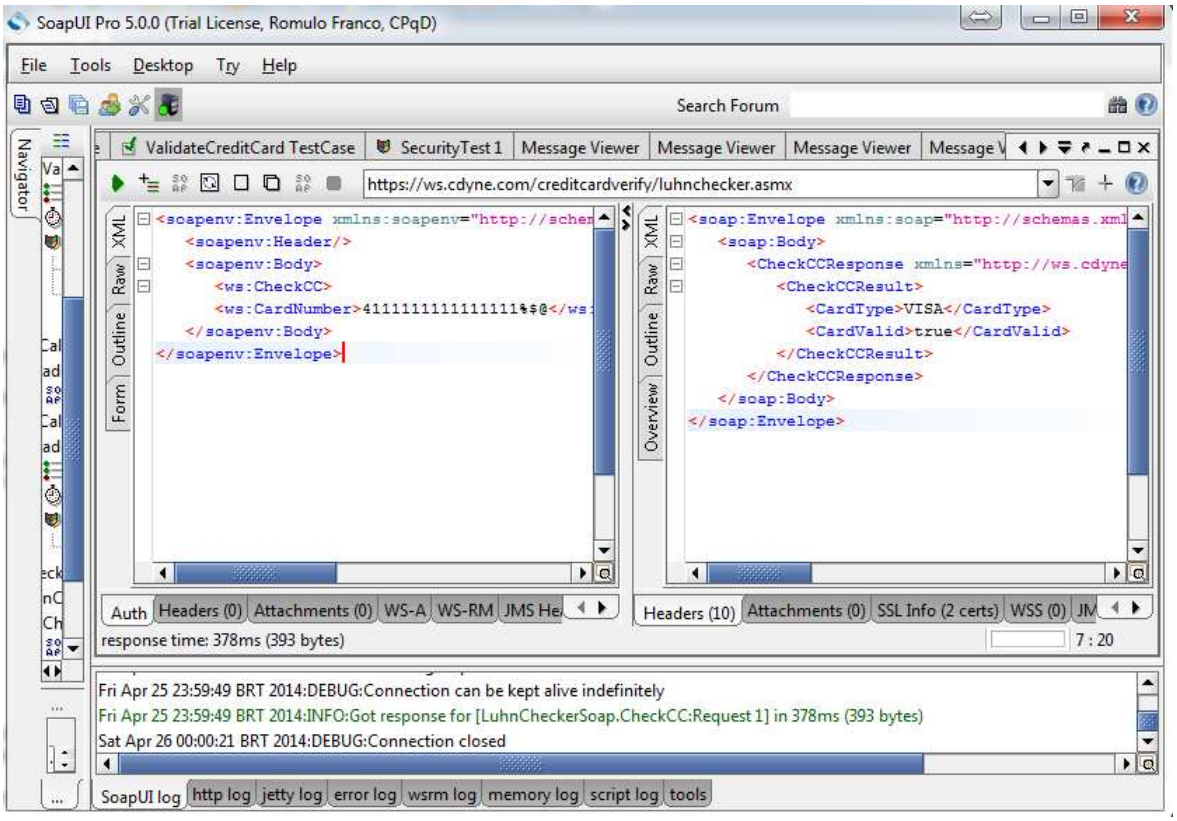



Figura 5.22: Teste do SW 2 - CheckCC com a confirmação da falha identificada na soapUI

Tabela 5.9: Resultado final sobre o atributo de QoS Robustez para os serviços

| Serviço Web | N.S. | N.V.E. | N.F. | F.P. |
|---|---|---|---|---|
| SW Interno | 9 | 1 | 0 | 2 |
| SW 1 - CreditCardValidationService | 9 | 1 | 0 | 1 |
| SW 2 - CheckCC | 6 | 1 | 1 | 1 |
| SW 3 - Validator | 6 | 0 | 0 | 0 |
| SW 4 - EmailValidationService | 6 | 1 | 0 | 1 |
| Legenda: N.S. - Núm. de scripts, N.V.E - Núm. de vulnerabilidades encontradas, N.F. - Núm. de falhas, F.P. - Falsos positivos | | | | |

1. **É possível detectar vulnerabilidades em serviços Web através da solução?**

   Durante a execução do estudo obtivemos resultados positivos e satisfatórios, onde foi possível identificar vulnerabilidades e uma falha grave em um dos serviços.

2. **Existe flexibilidade em executar os scripts de substituição definidos pelo usuário em ataques?**

   Durante a execução do estudo, inserimos scripts de forma manual para dois tipos de serviço, público e um serviço interno. Em ambos os casos, há flexibilidade e é possível executar diferentes ataques, incluindo ataques não relacionados neste estudo, porém, ataques por substituição na *workload*.

3. **Qual a precisão do diagnóstico emitido pela FlexMonitorWS ao identificar vulnerabilidades em serviços Web?**

   Ao executar o estudo, inserimos um serviço interno e primeiramente, executamos os scripts elencados para este serviço e posteriormente comparamos o resultado. Contudo, somente este comparativo não podemos assegurar com certeza uma resposta mais satisfatória, porque percebemos durante a execução do estudo, o comportamento do lado do servidor que está sob nosso domínio. Os demais serviços, podem ter comportamentos diferentes, principalmente nos resultados que contêm código de status 200.

Para assegurar uma resposta positiva e satisfatória, executamos um conjunto de scripts semelhantes na solução SoapUI. Ela permite executar uma gama muito maior de scripts de ataques e comparar os resultados obtidos nesta solução com nossos resultados.

A Figura 5.23 apresenta o resultado da execução de um conjunto de scripts utilizando a SoapUI. Para todos os serviços, o resultado foi semelhante ao nosso. Todavia, a SoapUI permite injetar um número muito maior de scripts e considerar outros tipos que não foram considerados no estudo de caso, por isso, o número de vulnerabilidades para o tipo *Malformed XML* é maior para cada serviço. No caso, como apresentado na Figura 5.23 foram três alertas de vulnerabilidades.

Como o resultado foi semelhante, cabe ressaltar que em ambas soluções podem gerar falsos positivos. Mas, a identificação da vulnerabilidade em ambas soluções conclui que o diagnóstico fornecido pela FlexMonitorWS está próximo às soluções existentes no mercado, para scripts do tipo *Malformed XML*.

Na Figura 5.25, apresentamos somente o resultado para um único serviço (SW 1) por questões de espaço. Repare na Figura 5.23 foram encontrados 3 vulnerabilidades para os scripts do tipo *Malformed XML* com a SoapUI, em contrapartida nenhuma vulnerabilidade foi encontrada para scripts do tipo *XML Bomb*.

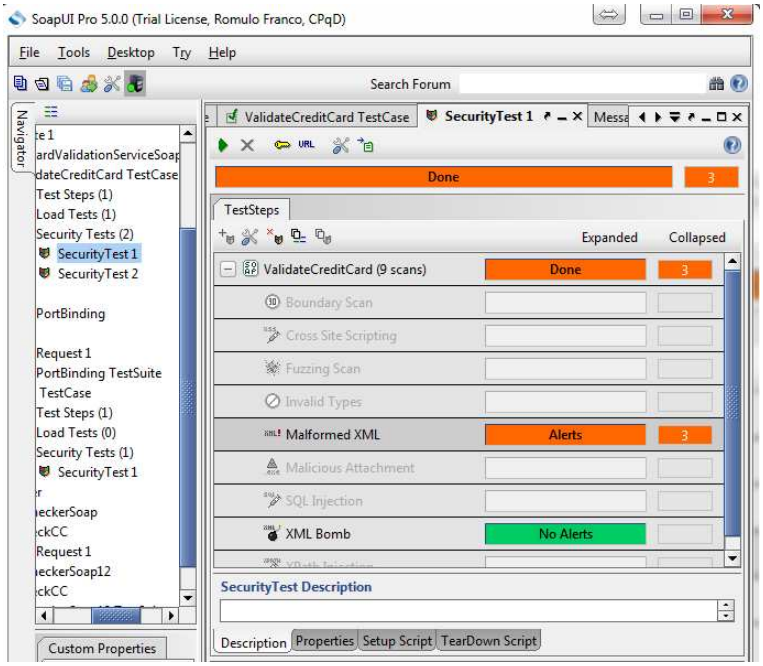



Figura 5.23: Teste do SW 1 - CreditCardValidationService com a confirmação dos alertas de vulnerabilidades identificadas na soapUI

Portanto, a hipótese endereçada por este estudo é verdadeira, pois a FlexMonitorWS pode ser considerada uma solução para injeção de falhas na requisição ao serviço.

## 5.4 Conclusões, lições aprendidas e limitações dos estudos de caso

Aqui traçamos as principais conclusões, limitações e dificuldades encontradas durante a execução dos estudos de caso.

### 5.4.1 Aplicação do cenário controlado utilizando máquinas virtualizadas

A criação do cenário controlado aplicado ao estudo de caso 1, foi complexa, pois envolve naturalmente sistemas distribuído. É da natureza de tais sistemas, como serviços Web, tal característica. Neste caso, foram utilizados três computadores, um deles um servidor para duas máquinas virtuais.

Máquinas virtualizadas no estudo de caso podem representar uma falha. Se a interrupção gerada impactar em mais de uma máquina virtualizada, embutidas em uma única máquina física, o resultado gerado pela interrupção poderá ter interferências. Aqui ponderamos a avaliação em máquinas virtuais e agimos com cautela em aplicar as interrupções somente nos casos onde as duas máquinas virtuais não entrariam em uma possível avaliação juntas.

### 5.4.2 Execução do cenário controlado e qualidade dos dados para análise

O estudo de caso 1 foi aplicado por mais de quatro vezes. Na primeira vez, surgiram problemas técnicos na solução e que somente foram detectados na análise do resultado posterior a um longo tempo de monitoração. Todo o resultado colhido foi desconsiderado na primeira execução. Corrigido o erro na solução foi retomado.

Na segunda tentativa, com todo o cenário re-preparado, iniciou-se a execução e percebeu que sem gerar interrupções, não seria possível detectar anomalias e não iria refletir problemas factíveis do mundo real, embora, algumas pequenas interferências normais do próprio ambiente geram pequenas, mas, poucas informações para aferir qualquer julgamento.

Na terceira tentativa, para poupar tempo, tentamos encaixar 12 horas de execução em uma escala de tempo de apenas 2 horas, gerando as interrupções em curtos intervalos de tempo. Os resultados não foram bons para esta tentativa também, pois muitos problemas em um curto espaço de tempo, tendo frequência de execução a cada um minuto, a qualidade dos dados foi ruim e resultou em uma análise com dados distorcidos de difícil compreensão. Então foi necessária efetuar uma quarta execução do estudo de caso, considerando um período de 14 horas, com interrupções espaçadas.

Durante as três primeiras tentativas houve um amadurecimento e um entendimento do cenário, as complicações existentes e como poderia ser contornado sem causar impacto no resultado do estudo de caso. Durante a quarta tentativa, os resultados foram colhidos e percebeu-se a melhora na qualidade dos dados em relação aos dados das tentativas anteriores, e foi possível analisar e cruzar os dados e estes dados resultantes é a razão principal do conteúdo do estudo de caso 1 e foi utilizado para análise da FlexMonitorWS.

### 5.4.3 Protocolos de rede TCP e ICMP durante a análise

Algumas características do ambiente podem interferir, razão deste fato é, por exemplo, a rede conectada por pontos físicos e alguns pontos através rede sem fio, é possível perceber no gráfico da Figura 5.24 (Monitoração dos pontos na rede) que o uso do ping que usa o protocolo ICMP, oferece resultados muito divergentes quando comparados com solução que usa outro protocolo (e.g. TCP). No ping, o protocolo usado não tem garantia de entrega, enquanto que, na monitoração por invocação, se faz uso do serviço por meio do protocolo TCP que contêm garantias de entrega. Portanto, vai garantir uma maior disponibilidade do serviço, mesmo quando há oscilações na rede. O cruzamento de informação resultante da monitoração da rede (ICMP) e do serviço (TCP) deve ser analisado com ponderação.

De um lado, uma possível saída seria oferecer solução que por padrão seja convencionado o uso de um único protocolo de comunicação. Por outro lado, a variedade de protocolos pode contribuir em entender de diferentes formas o comportamento do ambiente. Esta análise deve ficar a cargo do administrador da solução de monitoração. Uma melhor saída, seria acrescentar estas duas possibilidades como características da LPS e permitir que seja uma variabilidade da solução (utilizar monitoração por protocolo ICMP ou protocolo TCP).

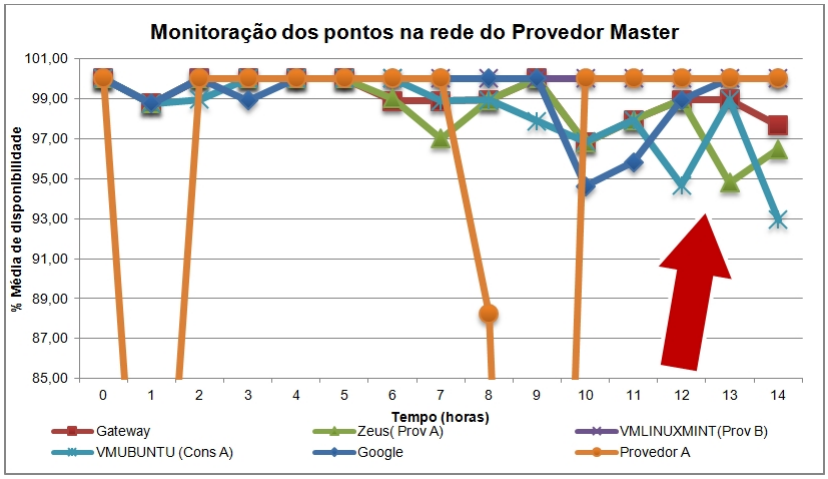



Figura 5.24: Monitoração dos pontos da rede

Note na Figura 5.24, a oscilação dos dados de disponibilidade entre as horas 10 até a hora 14, remete a falhas relativas à rede sem fio. Durante os pings em um dado IP podem ocorrer perdas de pacotes (exceder o tempo de resposta) e isso é contabilizado como indisponibilidade, baixando o percentual médio do atributo avaliado através da monitoração da rede. Já com o serviço Web do Provedor A isso não ocorre e a disponibilidade é alta.

### 5.4.4 Análise individual de atributos de QoS na massa de resultados

Cada atributo de QoS tem uma forma individual de análise. Durante a coleta dos dados, se o serviço está indisponível todos os atributos de QoS serão afetados. Isso ocorre, porque quando a monitoração colhe os dados, e o atributo de QoS está elencado para ser registrado ou notificado sobre seu comportamento, as amostras obtidas durante a monitoração resultará em um cálculo para obter o valor do atributo. Se o serviço está indisponível, para aquele período que ocorreu a monitoração, o dado registrado será -1, nulo ou não será registrado nenhum valor. Durante a análise, se não há nenhum dado, ou se o valor é -1, para aquele período há uma degradação na média para o atributo de QoS.

Uma saída para esta avaliação, é estudar o comportamento do atributo de QoS e efetuar uma monitoração de forma isolada, e não efetuando cálculos baseados nas médias, mas considerar um cálculo mais aprimorado.

### 5.4.5 Análise da massa de dados resultantes da monitoração

A monitoração durante um longo período de tempo e dependendo da sua frequência de execução e do número de alvos elencados, pode resultar em uma massiva quantidade de informações e que para a análise dependerá de técnicas sofisticadas. É necessária antecipar tais técnicas e agregar a solução de monitoração. Técnicas como mineração de dados pode ser uma saída.

Nos estudos de caso aplicados para a FlexMonitorWS, utilizamos planilhas eletrônicas para separar e agrupar os dados de diferentes maneiras, porém, é um trabalho que demanda tempo e esforço, em situações que demandam agilidade para obter uma informação valiosa da monitoração o tipo de técnica aplicada aqui pode ser um empecilho.

### 5.4.6 Estudo de caso 3 - Cenário Injeção de Falhas

A execução do estudo de caso 3 permitiu entender mais sobre as políticas de segurança que podem ser acrescentadas ao serviço Web. Contudo, serviços que implementam políticas do tipo *WS-ReliableMessaging* (WS-RM) e *WS-Addressing*, a execução do script é mais complexa, a FlexMonitorWS é limitada neste quesito, pois é necessário implementar as políticas em tempo de execução da invocação ao serviço.

### 5.4.7 Estudos primários da SLR e FlexMonitorWS

Durante SLR observamos que a maioria dos estudos primários avaliavam somente dois ou no máximo três atributos de QoS, isso pode ser decorrente da alta dificuldade em replicar um cenário do mundo real em um cenário controlado. É justificável uma análise envolvendo dois ou três atributos, pois isolar cada atributo de QoS demanda um alto

esforço, tempo, recursos físicos e de um paradigma organizacional bem estruturado. Este esforço gerado deve considerar o entendimento das várias e diferentes formas que um dado atributo tem em relação a pontos do ambiente, que inclui o serviço Web e a degradação que o ambiente oferece ao atributo.

Uma possível saída seria, criar *scripts* prontos que permitam redesenhar o cenário ou utilizar de ferramentas que possibilitam uma estratégia mais elaborada na criação de tais cenários. Como por exemplo, um dos estudos primários analisados faz uso de uma solução comercial para testes automatizados de carga denominada por *Mercury Load Runner* [Haiteng11].

## 5.5 Resumo do capítulo

Este capítulo abordou três estudos de caso. O primeiro visto na Seção 5.1 aborda um cenário controlado tentando reproduzir um cenário baseado no mundo real, através de um conjunto de interrupções. Este primeiro estudo de caso, teve como principal enfoque a viabilidade e flexibilidade da FlexMonitorWS em operar em composições de serviço, com diferentes alvos de monitoração. Os principais produtos gerados atenderam a mais de um perfil de monitoração, como por exemplo, monitorar serviços Web e monitorar rede. Outros perfis também foram atendidos durante a condução do primeiro estudo de caso.

No segundo estudo de caso, visto na Seção 5.2 aborda um cenário real, onde são inseridos quatro serviços Web públicos para serem monitorados. Tais serviços são reais e permitiram avaliar a viabilidade e capacidade da FlexMonitorWS em monitorar serviços Web públicos (SOAP). A monitoração aplicada do lado do consumidor permitiu detectar anomalias nestes serviços.

Em ambos os estudos de caso, foi possível avaliar a solução proposta em relação a sua viabilidade, aplicabilidade e flexibilidade em oferecer diferentes monitores o que também contribui com a reusabilidade da solução.

No terceiro estudo de caso, visto na Seção 5.3 a FlexMonitorWS foi aplicada para injetar falhas para diferentes tipos de ataques a serviços Web. Os resultados foram positivos e satisfatórios ao encontrar vulnerabilidades e falhas nos serviços elencados para o estudo.

Na Seção 5.4, foi elencado um conjunto de conclusões ligadas a dificuldades e limitações da FlexMonitorWS encontrados nos estudos de caso. Esta seção ajuda a entender possíveis saídas para aperfeiçoar a FlexMonitorWS.

No próximo capítulo são apresentadas as conclusões finais desse projeto de mestrado e destaca algumas contribuições e pontos de melhoria em trabalhos futuros.

# Capítulo 6

# Conclusões e trabalhos futuros

Neste capítulo apresentamos as conclusões desta dissertação, e em seguida discutimos sobre trabalhos futuros.

## 6.1 Conclusões

Este projeto teve por finalidade apresentar uma solução de monitoração que preenchesse lacunas encontradas na literatura de soluções de monitoração. A solução apresentada teve como foco a flexibilidade. A adoção de técnicas de Linhas de Produtos de Software para desenvolver uma família de monitores semelhantes, com capacidade de monitorar uma gama de elementos, ofereceria entendimento sobre problemas ocorridos em serviços Web que tem impacto na degradação de atributos de QoS. Nossa questão de pesquisa foi:

**RQ1:** Como aprimorar a flexibilidade das ferramentas de monitoração de atributos de qualidade de serviços a fim de melhor apoiar diferentes perfis de monitoração?

Como resposta a nossa RQ1, executamos três estudos de caso. Os dois primeiros estudos de caso, nos apoiou em entender os conceitos ligados a RQ1 e foi possível evidenciar de forma satisfatória que a FlexMonitorWS é flexível. O terceiro estudo de caso aplicado, amplia a quantidade de funcionalidades da solução, uma vez que, é possível injetar falhas em serviços Web para diferentes tipos de ataques.

A partir dos estudos de caso, relatamos aqui três principais pontos ligados a flexibilidade e ao custo investido sobre LPS:

1. **Diferentes perfis de monitoração**

   Durante as execuções dos estudos de caso, evidenciamos que a solução apoia a flexibilidade em atender diferentes perfis de monitoração. Primeiramente, por meio das configurações do modelo de característica, e em um segundo momento, através das interfaces, da abordagem utilizada por meio da Programação Orientada a Aspectos, e pelos produtos já existentes nesta primeira versão.

Os seguintes atributos de QoS, que são previstos na solução, descritos meios de obtê-los e verificados e validados por meio dos estudos de caso são: 1) disponibilidade de Serviços Web, 2) disponibilidade de pontos da rede, 3) desempenho de serviços Web e de pontos da rede, 4) confiabilidade, 5) acurácia, 6) robustez, 7) estado e condição do hardware do Servidor e 8) Falhas em arquivos de Log. Na Seção de Apêndices tem-se a avaliação para os atributos de QoS, Falhas em Arquivos de Log e Estado do hardware, provenientes da monitoração aplicada no estudo de caso 1.

Nos estudos de caso, evidenciamos também, duas importantes propriedades da FlexMonitorWS. O comportamento independente de plataforma da solução de monitoração, bem como, os diferentes alvos monitorados, como, servidor, aplicação servidora, rede e principalmente o serviço Web.

Portanto, a combinação dos diferentes atributos de QoS e os diferentes alvos de monitoração, atende a uma pluralidade de perfis de monitoração.

2. **Fácil manutenção**

   A FlexMonitorWS obedece a padrões e métodos propostos na Seção 2. Por utilizar padrões fundamentalmente baseado em LPS, o forte uso destes padrões, colaborou de forma satisfatória e contribui com a facilidade de manutenção.

3. **Alto investimento e retorno da LPS**

   No princípio da criação da LPS para cada componente que fez parte do modelo de característica da FlexMonitorWS, existiu um alto esforço de investimento no tempo e da complexidade existente para implementá-lo. Contudo, o retorno deste alto investimento se traduz por meio da combinação destes componentes para criar uma família de monitores. Esta relação pode ser melhor entendida pelas estatísticas geradas pela FeatureIDE, sobre a quantidade de produtos oferecidos pela FlexMonitorWS. Na Seção de Apêndice (Apêndice A.2) é possível a partir do modelo de características gerar **26648** monitores distintos.

## 6.2 Contribuições do projeto

1. **Uma Revisão Sistemática da Literatura (Capítulo 3)**

   Foi possível identificar a partir de 33 estudos primários, lacunas não preenchidas pelas soluções existentes na literatura. Este trabalho nos forneceu uma visão mais abrangente sobre sistemas de monitoração de serviços Web. O que permitiu assim, criar uma taxonomia destes tipos de sistemas, nos guiando para o resto do trabalho.

2. **Protótipo de uma solução de monitoração de serviços Web.**

Além da aplicação sobre serviços Web, este protótipo pode também ser utilizado por administradores de rede na identificação de vários problemas de recursos da infraestrutura de TI.

3. **Solução de injeção de falhas nas requisições de serviços Web de forma transparente.**

   Protótipo de uma solução que permite criar scripts de injeção de falhas (*faultload*) nas requisições ao serviço Web. Os scripts podem ser inseridos no arquivo de parâmetros (target.properties). O resultado da execução de cada script pode identificar vulnerabilidades ou falhas no serviço de acordo com o estudo de caso 3. A solução pode executar os scripts diretamente ao serviço Web e o relatório é imediato à execução, sem necessitar de infraestruturas como proxy. Além disso, a forma de implementação aplicada aqui, torna a injeção ativa pois é inserida na invocação ao serviço e realizado como teste.

4. **Aplicação do modelo AO-FArM para identificação de interesses transversais (Seção 5.3.2):**

   Ao aplicar o modelo AO-FArM foi possível identificar interesses transversais. Tais interesses foram mapeados nos modelos de características e em visões de características OA o que permitiu o mapeamento destas características para arquiteturas de LPS.

   O modelo AO-FArM foi desenvolvido por Tizzei (2012), e sua aplicação resultou de forma satisfatória, em entender melhor a relação de interesses transversais com a solução de monitoração.

5. **Publicações:**

   Trabalhos publicados no decorrer desta dissertação:

   - **Franco, R.J.**; Rubira, C.M.; Nascimento A.S. FlexMonitorWS: *Uma solução de monitoração de serviços Web com foco em atributos de QoS*. In Tools Session of the 5th Brazilian Congress on Software, CBSoft, 2014, Maceió, Alagoas

   - **Franco, R.J.**; Rubira, C.M.; *Uma Solução para Monitoração de Serviços Web Baseada em Linhas de Produtos de Software*. IX Workshop de Teses, Dissertações e Trabalhos de Iniciação Científica. IC-UNICAMP, 2014, Campinas, SP, Brazil.

## 6.3 Trabalhos futuros

Os seguintes trabalhos apontam para direções futuras são:

1. **FlexMonitorWS e *Cloud Computting***

Com este novo advento, se faz necessário mover a monitoração para as nuvens. Dada a relevância existente no atual mercado em enxugar custos com a infraestrutura de TI, a infraestrutura ainda é real embora não esteja fisicamente compartilhando um espaço com a organização, a necessidade e demanda em monitoração passa a ser então nas nuvens.

2. **Atualização automática e remota**

   Com o funcionamento nas nuvens passa a ter a necessidade de atualização automática e remota. Esta atualização deveria abranger Perfis de Monitoração, resultados da monitoração e configurações dos monitores.

3. **Seleção de protocolos de comunicação**

   De acordo com os estudos de caso, seria importante agregar ao modelo de característica a seleção de protocolos de rede e aplicação. Na Seção 5.4.3 foi descrito um aspecto importante da monitoração, usando protocolo TCP em comparação com ICMP. No modelo de característica, tais protocolos deveriam compor pares de protocolos, alternando as possibilidades no modelo para a composição de uma configuração da arquitetura da Linha de Produto, para um ou outro ou ambos protocolos.

4. **Monitorar Serviços Web REST**

   A FlexMonitorWS atualmente tem a limitação de monitorar serviços Web com protocolo SOAP, um dos desafios elencados na Revisão Sistemática da Literatura era a possibilidade de monitorar serviços Web usando o protocolo RESTful. A partir de uma rápida busca por termos utilizando o Google Trends é possível avaliar que tal protocolo tem tido um interesse maior em comparação com o protocolo SOAP.

5. **Monitoração de serviços Web dinâmica**

   Uma monitoração dinâmica tem capacidade de entender o ambiente e remover atributos de QoS do Perfil de monitoração que estão degradados, devido a efeitos do emprego da própria monitoração que está em execução. Este tipo de abordagem, permite fornecer a existência de conflitos entre dois ou mais atributos de QoS. Isto é, quando um determinado atributo de QoS sofre impacto em detrimento a um outro que ainda não está sendo monitorado, a monitoração notifica sobre. Neste caso, pode ser oferecida uma abordagem do tipo pré-ordem de execução, ou seja, um atributo de QoS é precedido por outro durante sua execução. O trabalho de Weider et. al. (2007) oferece uma explanação detalhada sobre este tipo de abordagem e pode ser utilizada em conjunto.

6. **Notificação a partir do uso de linguagens**

A notificação da monitoração não deve fornecer alarmes com alta regularidade, isso acarreta em a solução perder credibilidade se somente existe uma forma de notificar. A notificação deve ser pontual com graus específicos de severidade. Uma das formas, seria permitir a FlexMonitorWS no *target.properties* ter um parâmetro que seria uma expressão ou um comando semelhante a XPath [XPath13], ou a partir de linguagens como Baresi et. al 2008b aplica. Tais linguagens, forneceria uma forma de analisar a massa de dados resultante da monitoração e notificar a partir da expressão criada. Isso diminuiria notificações excessivas e criaria uma forma inteligente de notificar.

7. **Geração de relatórios e gráficos a partir de técnicas de mineração de dados**

   A monitoração coleta dados em excesso dependendo de vários fatores, como frequência de execução, número de alvos monitorados, número de atributos de QoS avaliados. Esta quantidade de elementos resulta em uma massiva quantidade de dados que se torna impraticável o uso de simples relatórios para identificar problemas existentes apontados pela monitoração. Uma abordagem baseada em técnicas de *data mining* pode oferecer dados sumarizados com capacidade para aprimorar a tomada de decisão, frente as falhas identificadas.

8. **Injeção de falhas em serviços Web que implementam políticas**

   Embora, a FlexMonitorWS permita identificar por meio da interceptação se um dado serviço implementa a política *WS-ReliableMessaging*, ela ainda não é capaz de injetar falhas em serviços Web que contêm esta política implementada. Políticas de segurança como *WS-ReliableMessaging*, alteram o modo de invocação ao serviço tradicional, para um modo que permite garantir a entrega de mensagens mesmo sob presença de falhas da rede, aplicação ou sistema. Injetar falhas nestes tipos de serviços Web, deveria acrescentar um modo de invocação diferente do aplicado aqui.

9. **Injeção de falhas na resposta do serviço Web**

   A FlexMonitorWS permite injetar falhas somente na requisição ao serviço Web, a implementação desse modo foi proveniente dos objetivos propostos aqui, que no caso era detectar a existência do atributo de QoS Robustez para os serviços, e o emprego de injeção de falhas na resposta não nos apoiaria. Porém, para ampliar as funcionalidades da solução, uma possível saída, acrescentar por meio de arquivo de parâmetros em qual momento deveria ocorrer a injeção, na requisição, na resposta ou considerar a injeção em ambos os lados. Esta funcionalidade também pode ser viável se aplicada ao modelo de característica e na arquitetura de linha de produto, este é um ponto de variação e tem estas duas variantes.

10. **Atributos de evolução e estudo empírico da LPS**

Em Tizzei (2012b) foi usado o atributo de evolução que é um indicador sobre a facilidade de evolução do sistema a partir da análise de diferentes métricas (e.g. acoplamento entre classes, coesão das classes, espalhamento de interesses e impacto de mudanças nas classes). A abordagem em seu estudo, se apoia em cenários de evolução, e coletou métricas aos atributos de evolução de arquitetura para cada versão da LPSs utilizada no estudo em questão. Nos estudos realizados por Tizzei (2012b), os melhores resultados foi a de orientação a aspectos.

Aqui, utilizamos abordagem semelhante para a implementação da FlexMonitorWS. Contudo, não foi aplicado um estudo empírico para avaliar quantitativa e qualitativamente a partir de diferentes métricas (e.g. entrelaçamento, espalhamento das características e dos pontos de variação, acoplamento entre módulos e impacto de mudanças) a facilidade de evolução da FlexMonitorWS.

11. **Inteface GUI**

    Por fim, não menos importante, por ter sido implementado um protótipo para o projeto, um dos casos de uso e uma das características do modelo preliminar vista na Seção 4.3.1 é a inclusão de um controle e gerenciamento da solução por meio de uma interface homem-máquina. Esta interface, permite aprimorar o gerenciamento da FlexMonitorWS, ao permitir gerenciar configurações de cada Perfil de monitoração e visualizar resultados por Perfil.

# Referências Bibliográficas


[Al-Masri09] E. Al-Masri and Q. H. Mahmoud. 2009. Web Service Discovery and Client Goals. Computer 42, 1 (January 2009), 104-107.

[Alonso04] G. Alonso, F. Casati, H. Kuno, and V. Machiraju. Web Services: Concepts, Architectures and Applications. Springer, Berlin, 2004

[Alves05] V. Alves, P. Matos Jr., L. Cole, P. Borba, and G. Ramalho. Extracting and evolving mobile games product lines. In LNCS, volume 3714/2005, 2005.

[Anastasopoules01] C. Gacek and M. Anastasopoules. Implementing product line variabilities. SIGSOFT Softw. Eng. Notes , 26(3), 2001.

[Anastasopoulos04] M. Anastasopoulos and D. Muthig. An evaluation of aspect-oriented programming as a product line implementation technology. In ICSR, pages 141–156, 2004.

[Araban04] S. Araban and L. Sterling, "Measuring Quality of Service for Contract Aware Web-Services,"in 1st Australian Workshop on Engineering Service-Oriented Systems (AWESOS 2004). Melbourne, Australia, 2004

[Artaiam08] N. Artaiam and T. Senivongse. Enhancing Service side QoS monitoring for web services. In Proceedings of the 2008 Ninth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, Washington, DC, USA, 2008. IEEE Computer Society.

[Atkinson02] C. Atkinson, J. Bayer, C. Bunse, E. Kamsties, O. Laitenberger, R. Laqua, D. Muthig, B. Paech, J. Wüst, J. Zettel, Component-based Product Line Engineering with UML, Addison-Wesley, Boston, MA, USA, 2002.

[Baelen00] S. Van Baelen, D. Urting, W. Van Belle, V. Jonckers, T. Holvo et, Y. Berbers, and K. De Vlaminck. Toward a unified terminology for component-based development. In Fifth international workshop on component-oriented programming (WCOP 2000) , Canes, France, June 2000.

[Baresi08] L. Baresi, S. Guinea, and L. Pasquale. 2008. Towards a unified framework for the monitoring and recovery of BPEL processes. In Proceedings of the 2008 workshop on Testing, analysis, and verification of web services and applications (TAV-WEB '08), Tevfik Bultan and Tao Xie (Eds.). ACM, New York, NY, USA, 15-19.

[Baresi08a] L. Baresi , S. Guinea, L. Pasquale, 2008. Integrated and Composable Supervision of BPEL Processes. In Proceedings of the 6th International Conference on Service-Oriented Computing(ICSOC '08)

[Baresi08b] L. Baresi and S.Guinea. 2008. A dynamic and reactive approach to the supervision of BPEL processes. In Proceedings of the 1st India software engineering conference (ISEC '08). ACM, New York, NY, USA, 39-48.

[Baresi09] L. Baresi, S. Guinea, M. Pistore, M. Trainotti: Dynamo + Astro: An Integrated Approach for BPEL Monitoring. ICWS 2009: 230-237

[Baresi10] L. Baresi, S. Guinea, O. Nano, G. Spanoudakis, "Comprehensive Monitoring of BPEL Processes,"IEEE Internet Computing, vol. 14, no. 3, pp. 50-57, May-June 2010

[Bayer99] Bayer, J.; Flege, O.; Knauber, P.; Laqua, R.; Muthig, D.; Schmid, K.; Widen, T.; Debaud, J.-M. PuLSE: a methodology to develop software product lines. In: Proceedings of the Symposium on Software reusability, 1999, Los Angeles, U.S.A. New York, U.S.A.: ACM. p. 122–131.

[Benbernou10] S. Benbernou, I. Brandic, C. Cappiello, M. Carro, M. Comuzzi, A. Kertész, K. Kritikos, M. Parkin, B. Pernici and P. Plebani. 2010. Modeling and negotiating service quality. In Service research challenges and solutions for the future internet, Mike Papazoglou, Klaus Pohl, Michael Parkin, and Andreas Metzger (Eds.). Springer-Verlag, Berlin, Heidelberg 157-208

[Benavides07] D. Benavides, S. Segura, P. Trinidad, A. Ruiz-Cortés, FAMA: tooling a framework for the automated analysis of feature models, in: Proceeding of the First International Workshop on Variability Modelling of Software-intensive Systems (VAMOS), 2007, pp. 129-134.

[Berg05] Berg, K.; Bishop, J.; Muthig, D. Tracing software product line variability: from problem to solution space. In: Proceedings of the Annual research Conference of the South African institute of computer scientists and information technologists on IT research in developing countries, 2005, Republic of South Africa. Republic of South Africa: South African Institute for Computer Scientists and Information Technologists. p. 182–191

[Breivold07] H. P. Breivold and M. Larsson. Component based and service oriented software engineering: Key concepts and principles. In Proceedings of the 33rd EUROMICRO Conference on Software Engineering and Advanced Applications, Washington, DC, USA, 2007. IEEE Computer Society.

[Bubak08] O. Bubak and H. Gomaa. Applying software product line concepts in service orientation. Int. J. Intel l. Inf. Database Syst. , 2, November 2008.

[Cabrera12] Cabrera, O.; Franch, X.; , "A quality model for analysing web service monitoring tools,"Research Challenges in Information Science (RCIS), 2012 Sixth International Conference on , vol., no., pp.1-12, 16-18 May 2012

[Campbell08] G. H. Campbell. Renewing the product line vision. In Proceedings of the 2 008 12th International Software Product Line Conference, Washington, DC, USA, 2008. IEEE Computer Society

[Capilla05] R. Capilla and N. Y. Topaloglu. Product lines for supporting the composition and evolution of service oriented applications. In Proceedings of the Eighth International Workshop on Principles of Software Evolution , Washington, DC, USA, 2005. IEEE Computer Society.

[Cachin00] Cachin C, Camenisch J, Dacier M, Deswarte Y, Dobson J, Horne D, et al. MALICIOUS and Accidental-Fault Tolerance in Internet Applications: Reference Model and Use Cases - Relatório técnico. MAFTIA. University of Newcastle Upon Tyne LAAS report no. 00280, Project IST-1999-11583; Aug 2000.

[Chang07] S. H. Chang and S. D. Kim. A variability modeling method for adaptable services in service-oriented computing. In SPLC '07: Proceedings of the 11th International Software Product Line Conference, pages 261-268, Washington, DC, USA, 2007. IEEE Computer Society.

[Cheesman00] Cheesman, J.; Daniels, J. UML components: a simple process for specifying component-based software. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2000

[Clements01] Clements, P. and Northrop, L. (2001) Software product lines: practices and patterns. Addison Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

[Comes12] D. E. Comes, H. Baraki, R. Reichle, K. Geihs: BPRules and the BPR-Framework: Comprehensive Support for Managing QoS in Web Service Compositions. 12th IFIP International Conference on Distributed Applications and Interoperable Systems - June 2012. Pages 222-235

[Delgado04] N. Delgado, A. Q. Gates and S. Roach. A Taxonomy and Catalog of Runtime Software-Fault Monitoring Tools. IEEE Transactions on software Engineering, pages 859-872, December, 2004.

[Dias10] M. Dias, L. Tizzei, C. Rubira, A. Garcia and J. Lee. Leveraging aspect-connectors to improve stability of product-line variabilities. In Proceedings of the VaMoS'2010 -Workshop on Variability Modelling of Software-intensive Systems, pages 21–28, 2010.

[Dustdar05] S. Dustdar and W. Schreiner. A survey on web services composition. Int. J. Web Grid Serv., 1(1), 2005.

[Fakhfakh08] K. Fakhfakh, T. Chaari, S. Tazi, K. Drira, and M. Jmaiel. 2008. A Comprehensive Ontology-Based Approach for SLA Obligations Monitoring. In Proceedings of the 2008 The Second International Conference on Advanced Engineering Computing and Applications in Sciences (ADVCOMP '08). IEEE Computer Society, Washington, DC, USA, 217-222

[Fei08] L. Fei, Y. Fangchun, S. Kai, and S. Sen. 2008. A Policy-Driven Distributed Framework for Monitoring Quality of Web Services. In Proceedings of the 2008 IEEE International Conference on Web Services (ICWS '08). IEEE Computer Society, Washington, DC, USA, 708-715

[Figueiredo08] E. Figueiredo, N. Camacho, C. S, M. Monteiro, U. Kulesza, A. Garcia, S. Soares, F. Ferrari, S. Khan, F. Filho, and F. Dantas. Evolving software product lines with aspects: an empirical study on design stability. In ICSE, 2008.

[Garlan00] D. Garlan, R. T. Monro e, and D. Wile. Foundations of component based systems. chapter Acme: architectural description of component-based systems. Cambridge University Press, New York, NY, USA, 2000

[Garlan00] Garlan, D.; Monroe, R.; Wile, D. Acme: Architectural description of component-based systems. In: LEAVENS, G. T.; Sitaraman, M. (Eds.) Foundations of Component-Based Systems. Cambridge University Press, 2000. Cap. 3, p. 47 – 68.

[Gayard08] Gayard, L. A.; Rubira, C. M. F.; de Castro Guerra, P. A. COSMOS*: a COmponent System MOdel for Software Architectures. Technical Report IC-08-04, Instituto de Computação, UNICAMP, 2008

[Geraci91] A. Geraci, IEEE standard computer dictionary: Compilation of IEEE standard computer glossaries. Piscataway, NJ, USA: IEEE Press, 1991.

[Gmach08] Daniel Gmach, Stefan Krompass, Andreas Scholz, Martin Wimmer, and Alfons Kemper. 2008. Adaptive quality of service management for enterprise services. ACM Trans. Web 2, 1, Article 8 (March 2008), 46 pages.

[Godse10] M. Godse, U. Bellur, and R. Sonar. Automating QoS based service selection. In Proceedings of the 2010 IEEE International Conference on Web Services, ICWS '10, Washington, DC, USA, 2010. IEEE Computer Society.

[Goel10] N. Goel and R.K. Shyamasundar, Automatic Monitoring of SLAs of Web Services, Services Computing Conference (APSCC), 2010 IEEE Asia-Pacific

[Gomaa02] H. Gomaa and M. E. Shin. Multipleview metamodeling of software product lines. In ICECCS '02: Proceedings of the Eighth International Conference on Engineering of Complex Computer Systems , Washington, DC, USA, 2002. IEEE Computer Society.

[Gomaa04] H. Gomaa. Designing Software Product Lines with UML: From Use Cases to Pattern-Based Software Architectures. Addison Wesley Longman Publishing Co., Inc., Redwo o d City, CA, USA, 2004.

[Gomaa05] H. Gomaa and M. Saleh. Feature driven dynamic customization of software product lines. In ICSR , 2006.

[Griswold06] Griswold, W.; Shonle, M.; Sullivan, K.; Song, Y.; Tewari, N.; Cai, Y.; Rajan, H. Modular software design with crosscutting interfaces. Software, IEEE, v. 23, n. 1, p. 51 –60, 2006.

[Gunther98] Gunther, N. J. 1998. The Practical Performance Analyst, published by McGraw-Hill.

[Guo08] N. Guo, T. Gao, and B. Zhang. 2008. Trusted Assessment of Web Services Based on a Six-Dimensional QoS Model. In Proceedings of the 2008 IEEE Asia-Pacific Services Computing Conference (APSCC '08). IEEE Computer Society, Washington, DC, USA, 123-127.

[Gurp01] J. V. Gurp, J. Bosch, and M. Svahnb erg. On the notion of variability in software product lines. In Proceedings of the Working IEEE/IFIP Conference on Software Architecture, WICSA '01, Washington, DC, USA, 2001. IEEE Computer Society.

[Haiteng11] Z. Haiteng, S. Zhiqing, Z. Hong: Runtime Monitoring Web Services Implemented in BPEL. Uncertainty Reasoning and Knowledge Engineering (URKE), 2011 International Conference

[Halima08] R. B. Halima, K. Guennoun , M. Jmaiel, and K. Drira. Non-intrusive QoS Monitoring and Analysis for Self-Healing Web Services. Applications of Digital Information and Web Technologies, 2008. ICADIWT 2008. First International Conference on the Applications of Digital

[Huhns05] M. N. Huhns and M. P. Singh. Service-oriented computing: Key concepts and principles. IEEE Internet Computing, 9(1), 2005.

[Iso01] International Organization for Standarization, ISO/IEC standard 9126: Software Engineering – Product Quality, part 1. 2001.

[Ivanovic10] D. Ivanovic & M. Carro (2010). An Initial Proposal for Data-Aware Resource Analysis of Orchestrations with Applications to Predictive Monitoring. International Workshops, ICSOC/ServiceWave 2009, Revised Selected Papers. Lecture Notes in Computer Science, Vol. 6275, (6275). Springer

[Jayanti05] V. B. Kumar Jayanti, M. Hadley, SOAP with Attachments API for Java™ (SAAJ) 1.3, 2005, Sun Microsystems, Inc.

[Kang98] Kang, K. C.; Kim, S.; Lee, J.; Kim, K.; Shin, E.; Huh, M. FORM: A feature oriented reuse method with domain-specific reference architectures. Annals of Software Engineering, v. 5, p. 143–168, 1998

[Kastner09] C. Kastner, T. Thum, G. Saake, J. Feigenspan, T. Leich, F. Wielgorz, and S. Apel. 2009. FeatureIDE: A tool framework for feature-oriented software development. In Proceedings of the 31st International Conference on Software Engineering (ICSE '09). IEEE Computer Society, Washington, DC, USA, 611-614.

[Katsaros11] G. Katsaros, R. Kubert, and G. Gallizo. 2011. Building a Service-Oriented Monitoring Framework with REST and Nagios. In Proceedings of the 2011 IEEE International Conference on Services Computing (SCC '11). IEEE Computer Society, Washington, DC, USA, 426-431.

[Kiczales01] G. Kiczales, E. Hilsdale, J. Hugunin, M. Kersten, J. Palm, and W. G. Griswold. An overview of aspectj. In ECOOP '01: Proceedings of the 15th European Conference on Object-Oriented Programming, pages 327–353, London, UK, 2001. Springer-Verlag.

[Kiczales97] Kiczales, G.; Lamping, J.; Mendhekar, A.; Maeda, C.; Lopes, C.;Marc Loingtier, J.; IRWIN, J. Aspect-oriented programming. In: Proceedings of the European Conference on Object-Oriented Programming, 11., 1997, Jyväskylä,Finland. Springer. p. 220–242

[Kim06] Y.G. Kim, J.W. Kim, S.O. Shin, and D.K. Baik. Managing variability for software product line. In SERA '06: Proceedings of the Fourth International Conference on Software Engineering Research, Management and Applications, Washington, DC, USA, 2006. IEEE Computer Society.

[Kitchenham04] B. Kitchenham. Procedures for performing systematic reviews. Technical Report Technical Report TR/SE-0401, Keele University and NICTA, 2004.

[Krueger02] C. W. Krueger. Variation management for software production lines. In SPLC 2: Proceedings of the Second International Conference on Software Product Lines , pages 37-48, London, UK, 2002. SpringerVerlag.

[Krueger02a] C. W. Krueger. Easing the transition to software mass customization. In International Workshop on Software Product-Family Engineering, pages 282-293. Springer, 2002

[Leitner09] P. Leitner, B. Wetzstein, F. Rosenberg, A. Michlmayr, S. Dustdar, and F. Leymann. 2009. Runtime prediction of service level agreement violations for composite services. In Proceedings of the 2009 international conference on Service-oriented computing (ICSOC/ServiceWave'09)

[Linden07] F. J. van der Linden, K. Schmid, and E. Rommes. Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering. SpringerVerlag New York, Inc., Secaucus, NJ, USA, 2007.

[Liu10] A. Liu, Q. Li, L. Huang, and M. Xiao. Facts: A framework for fault-tolerant composition of transactional web services. IEEE Trans. Serv. Comput., 3, January 2010

[Ludwig03] H. Ludwig. Web services QoS: external SLAs and internal policies or: how do we deliver what we promise? Fourth International Conference on Web Information Systems Engineering Workshops. 4:115–120, Dec. 2003.

[Marzolla10] Marzolla, M.; Mirandola, R. 2010. QoS Analysis for Web Service Applications: a Survey of Performance-oriented Approaches from an Architectural Viewpoint. Technical Report UBLCS-2010-05 - February 2010

[Menasce02] D. A. Menascé. 2002. QoS Issues in Web Services. IEEE Internet Computing 6, 6 (November 2002), 72-75.

[Michlmayr09] A. Michlmayr, F. Rosenberg, P Leitner, and S. Dustdar. 2009. Comprehensive QoS monitoring of Web services and event-based SLA violation detection. In Proceedings of the 4th International Workshop on Middleware for Service Oriented Computing (MWSOC '09). ACM, New York, NY, USA, 1-6

[Moser08] O. Moser, F. Rosenberg, and S. Dustdar. 2008. Non-intrusive monitoring and service adaptation for WS-BPEL. In Proceedings of the 17th international conference on World Wide Web (WWW '08). ACM, New York, NY, USA, 815-824

[Muller12] C. Müller, M. Oriol, M. Rodríguez, X. Franch, J. Marco, M. Resinas, A. Ruiz-Cortés, "SALMonADA: A platform for Monitoring and Explaining Violations of WS-Agreement-compliant Documents", In Proceedings of the 4th International Workshop on Principles of Engineering Service-Oriented Systems, PESOS 2012, pp. 43-49, IEEE, June 2012.

[Noda08] Noda, N.; Kishi, T. Aspect-oriented modeling for variability management. In: Proceedings of the International Software Product Line Conference, 12., 2008, Limerick, Ireland. p. 213 –222.

[Papazoglou06] M. P. Papazoglou, P. Traverso, S. Dustdar, F. Leymann, and B. J. Kramer. Service-oriented computing research roadmap. In Dagstuhl Seminar Proceedings 05462 , April 2006.

[Papazoglou07] M. P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann. Service-oriented computing: State of the art and research challenges. Computer, 40, 2007.

[Pertet05] S. Pertet and P. Narasimhan, Causes of Failure in Web Applications (CMU-PDL-05-109), Technical Report, Carnegie Mellon University, 2005

[Pohl05] Pohl, K.; Böckle, G.; Van Der Linden, F. Software product line engineering: Foundations, principles, and techniques. Berlin/Heidelberg: Springer, 2005.

[Pressman06] Pressman, Roger S.;Engenharia de Software, 6a. ed. - São Paulo, McGraw-Hill, 2006, pp. 350-351.

[Props13] Properties, The Java Tutorails, Oracle, Acessado http://docs.oracle.com/javase/tutorial/essential/environment/properties.html em 25/nov/2013.

[Raimondi08] F. Raimondi, J. Skene, and W. Emmerich. 2008. Efficient online monitoring of web-service SLAs. In Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering (SIGSOFT '08/FSE-16). ACM, New York, NY, USA, 170-180.

[Ran03] S. Ran, "A Model for Web Services Discovery with QoS,"ACM SIGecom Exchanges, vol. 4, pp. 1-10, 2003

[Robillard07] Robillard, M. P.; Murphy, G. C. Representing concerns in source code. ACM Transactions Software Engineering Methodology, Canada, v. 16, n. 1, p. 3, 2007.

[Romano11] Romano, L., Mari, D.D., Jerzak, Z., Fetzer, C.: A novel approach to qos monitoring in the cloud. In: 1st International Conference on Data Compression, Communication and Processing. IEEE Computer Society Press (June 2011)

[Rosenberg09] F. Rosenberg, 2009. QoS-Aware Composition of Adaptive Service-Oriented Systems, Tese de doutorado, Technische Universität Wien, Viena, Austria.

[Runeson09] P. Runeson and M. Höst. 2009. Guidelines for conducting and reporting case study research in software engineering. Empirical Softw. Engg. 14, 2 (April 2009), 131-164.

[Salas12] Salas, Marcelo Invert Palma, 2012 - Metodologia de testes de segurança para análise de robustez de Web Services por injeção de falhas. UNICAMP, 2012, Dissertação de mestrado.

[Schoene09] T. Schoene, F. Kaspar, and C. Reich. Using WSDM and Web Service Ping for QoS based Web Service Selection. ECEASST , 17, 2009.

[Schroeder95] B.A. Schroeder, "On-Line Monitoring: A Tutorial," Computer, vol. 28, no. 6, pp. 72-78, June 1995.

[SEISPLv5] Software engineering institute. Framework for Software Product Line Practice versão 5.0 http://www.sei.cmu.edu/productlines/framework.html.

[Serhani10] Serhani, M.A. Al-Qirim, N. Benhareef, A. Enterprise Services (Business) Collaboration Using Portal and Soa-Based Semantics, 4th IEEE International Conference on Digital Ecosystems and Technologies (IEEE DEST 2010)

[Sidibe08] M. Sidibe, A. Mehaoua, QoS monitoring for end-to-end heterogeneous networks configurations management.Automation, Quality and Testing, Robotics, 2008. AQTR 2008. IEEE International Conference on;

[Sigar10] R. Morgan, D. MacEachern, Hyperic SIGAR SIGAR - System Information Gatherer And Reporter, 2010. Acessado https://support.hyperic.com/display/SIGAR/Home 22/nov/2013.

[Silva03] Silva JR., M. C. D.; Guerra, P. A. D. C.; Rubira, C. M. F. A java componente model for evolving software systems. In: Proceedings of the Automated Software Engineering Conference, 2003, Montreal, Canada. p. 327–330.

[Sinnema04] M. Sinnema, S. Deelstra, J. Nijhuis, and J. Bosch. Covamof: A framework for modeling variability in software product families. In Software Product Lines, Third International Conference, SPLC 2004, Boston, MA, USA, August 30, September 2, 2004, Proceedings, volume 3154 of Lecture Notes in Computer Science. Springer, 2004.

[Sinnema07] M. Sinnema and S. Deelstra. Classifying variability modeling techniques. Inf. Softw. Technol., 49, July 2007.

[Sochos06] Sochos, P.; Riebisch, M.; Philippow, I. The feature-architecture mapping (FArM) method for feature-oriented development of software product lines. In: Proceedings of the Annual IEEE International Symposium and Workshop on Engineering of Computer Based Systems, 13., 2006, Potsdam, Germany. IEEE Computer Society. p. 308–318.

[Sommerville95] I. Sommerville. Software Engineering. AddisonWesley, 5th edition, 1995

[Souza11] F. Souza, D. Lopes, K. Gama, N. S. Rosa, and R. Lima. Dynamic event-based monitoring in a soa environment. In OTM Conferences (2) , 2011.

[Spillner09] J. Spillner, A. Schill, Dynamic SLA Template Adjustments Based on Service Property Monitoring. Cloud Computing, 2009. CLOUD '09. IEEE International Conference on

[Svahnberg05] M. Svahnberg, J. V. Gurp, and J. Bosch. A taxonomy of variability realization techniques: Research articles. Softw. Pract. Exper., 35(8), 2005

[Tcpdump013] TCPDUMP, http://www.tcpdump.org/manpages/tcpdump.1.html. Acessado em 15/nov/2013.

[Thio05] N. Thio and S. Karunasekera. Automatic measurement of a qos metric for web service recommendation. In Proceedings of the 2005 Australian conference on Software Engineering, Washington, DC, USA, 2005. IEEE Computer Society

[Tian04] M. Tian, A. Gramm, H. Ritter, and J. Schiller. Efficient selection and monitoring of qos-aware web services with the ws-qos framework. In Proceedings of the 2004 IEEE/WIC/ACM International Conference on Web Intel ligence, WI'04, pages 152158, Washington, DC, USA, 2004. IEEE Computer Society.

[Tian10] H. Tian, X. Dong, X. Zhao, L. Zeng, X. Zhang. A novel SLA-driven verification and planning framework for QoS.Information Management and Engineering (ICIME), 2010 The 2nd IEEE International Conference on: Pages: 159-164

[Tizzei11] Tizzei, L.P. & C.M.F. Rubira. Aspect-Connectors to Support the Evolution of Component-Based Product Line Architectures: A Comparative Study . In Crnkovic, I. V. Gruhn e M. Books (eds.): Software Architecture, vol. 6903 de Lecture Notes in Computer Science, pags. 59-66. Springer, 2011.

[Tizzei12a] Tizzei, L.P, C.M.F. Rubira & J. Lee. An Aspect-based Feature Model for Architecting Component Product Lines. 38th Euromicro Conference on Software Engineering and Advanced Applications, 2012, Cesme, Turquia.

[Tizzei12b] Tizzei L. P., Evolução de Arquiteturas de Linhas de Produtos baseadas em Componentes e Aspectos, Tese de doutorado, Instituto de Computação, UNICAMP, 2012

[Truong06] H. Truong, R. Samborski, and T. Fahringer. 2006. Towards a Framework for Monitoring and Analyzing QoS Metrics of Grid Services. In Proceedings of the Second IEEE International Conference on e-Science and Grid Computing (E-SCIENCE '06). IEEE Computer Society, Washington, DC, USA, 65.

[Voelter07] Voelter, M.; Groher, I. Product line implementation using aspect-oriented and model-driven software development. In: International Software Product Line Conference, 11., 2007, Kyoto, Japan. p. 233 –24

[W3C03] W3C. QoS for Web Services: Requirements and Possible Approaches, W3C Working Group Note 25 November 2003

[Wang09] Q. Wang, J. Shao, F. Deng, Y. Liu, M. Li, J. Han, H. Mei, "An Online Monitoring Approach for Web Service Requirements,"IEEE Transactions on Services Computing, vol. 2, no. 4, pp. 338-351, 2009

[Weider07] Weider D. Yu, Rachana B. Radhakrishna, Sumana Pingali, and Vijaya Kolluri. 2007. Modeling the Measurements of QoS Requirements in Web Service Systems. Simulation 83, 1 (January 2007), 75-91.

[Weider07] Weider D. Yu, Rachana B. Radhakrishna, Sumana Pingali, and Vijaya Kolluri. 2007. Modeling the Measurements of QoS Requirements in Web Service Systems. Simulation 83, 1 (January 2007), 75-91.

[Wetzstein09] B. Wetzstein, P. Leitner, F. Rosenberg, I. Brandic, S. Dustdar, F. Leymann: Monitoring and Analyzing Influential Factors of Business Process Performance. EDOC 2009: 141-150

[WSInject10] Valenti AW, Maja WY, Martins E, Bessayah F, Cavalli A. WSInject: A Fault Injection Tool for Web Services - Relatório técnico, IC-UNICAMP. Campinas, Brasil, 2010.

[Yousefipour10] Yousefipour, A., Mohsenzadeh M., A New Broker-based Semantic Web Service Discovery Framework for Selecting and Ranking Suggested Web Services. ICCP '10 Proceedings of the Proceedings of the 2010

[Yu08] Q. Yu, X. Liu, A. Bouguettaya, and B. Medjahed. 2008. Deploying and managing Web services: issues, solutions, and directions. The VLDB Journal 17, 3 (May 2008), 537-572.

[Zadeh10] Zadeh ,M.H; Seyyedi ,M.A, "Qos Monitoring for Web Services by Time Series Forecasting", 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT), 2010.

[Zhang08] Zhang, J.; Cai, X.; Liu, G. Mapping features to architectural components in aspect-oriented software product lines. In: International Conference on Computer Science and Software Engineering, 2008, Wuhan, China. p. 94 – 97

# Apêndice A

## A.1 Estudo de caso 1: Cenário Controlado - Outros dados

Na Figura A.1 dados resultantes da monitoração de um arquivo de Log da aplicação Servidora em execução para o estudo de caso 1. Note na Figura A.1, os erros ou falhas detectadas representam as falhas geradas através da segunda interrupção do estudo de caso 1.

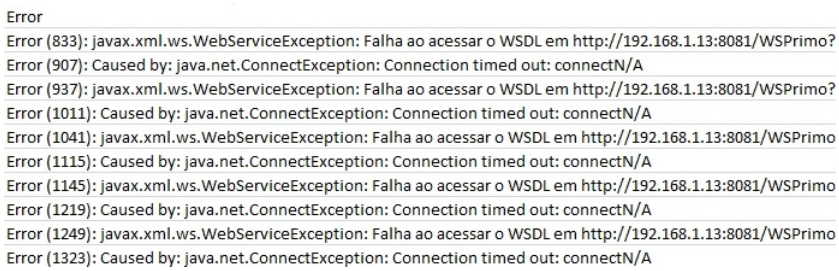



Figura A.1: Monitoração da Aplicação Servidora – Parte do Arquivo de Logs

A Figura A.2 apresenta resultados da monitoração sobre os recursos físicos do Servidor, especificamente sobre a memória *Swap* ou virtual, é um tipo de memória secundária usada pelo Sistema operacional para cache. Na Figura A.2, é apresentada a memória *Swap* durante o tempo de monitoração. A distribuição é dividida por memória usada (*Used*), memória livre (*Free*) e memória total.

Na Figura A.3, são apresentados resultados sobre a monitoração dos recursos físicos do Servidor (Provedor A) especificamente do percentual de uso do processador durante o tempo de monitoração. Na Figura A.3, tem a divisão, sendo *Idle* o percentual em que o processador ficava inativo. *User*, o percentual usado por aplicações do usuário, e *System*, percentual que aplicações do próprio sistema operacional usou no processamento durante o tempo de monitoração.

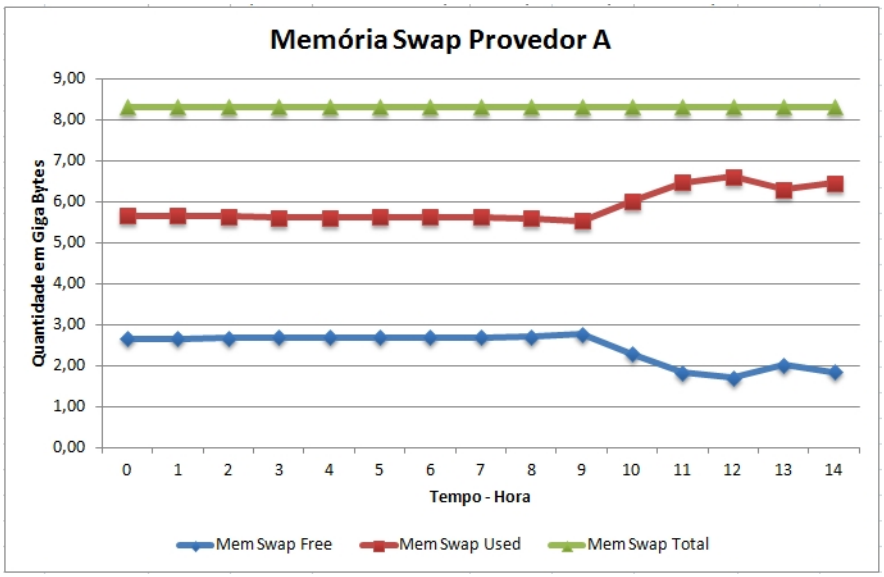



Figura A.2: Distribuição da memória *Swap* do Provedor A

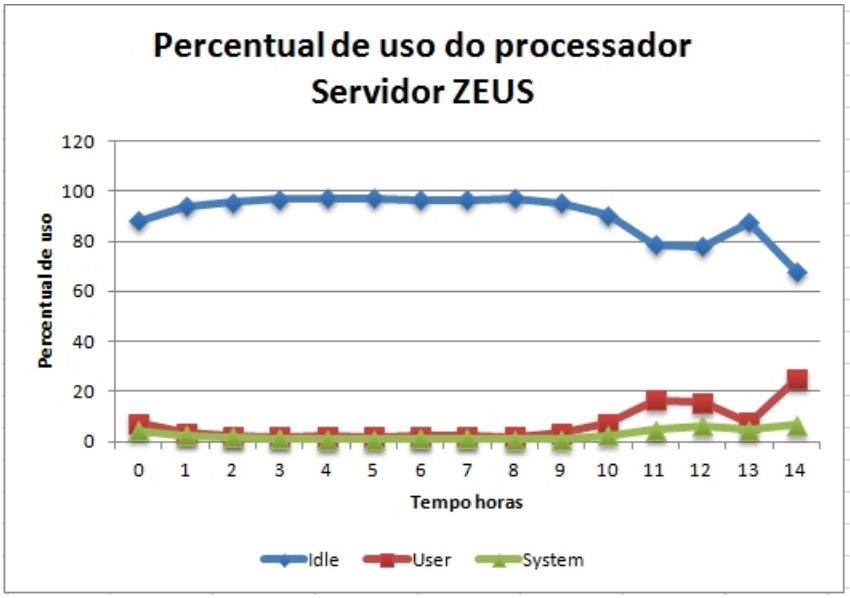



Figura A.3: Percentual de uso do processador no Provedor A

## A.2 Estatísticas do modelo

Ao executar a ferramenta FeatureIDE inserindo as características de acordo com o modelo proposto, o número total de produtos foi de **26648**. Outras estatísticas podem ser obtidas também e conferidas na Figura A.4.

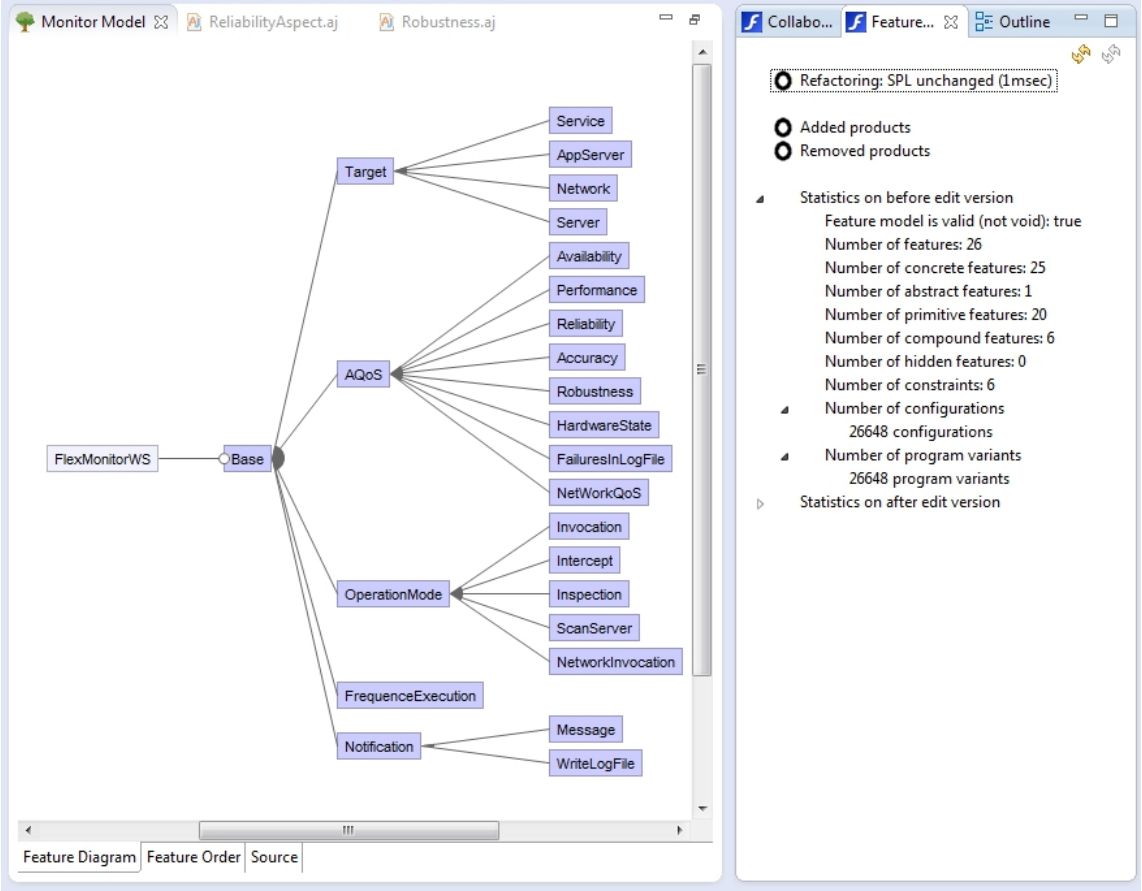



Figura A.4: Estatísticas geradas por meio da FeatureIDE