Detecção de Violações de SLA em Coreografias de Serviços Web

Victoriano Alfonso Phocco Diaz

DISSERTAÇÃO APRESENTADA
AO
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA
DA
UNIVERSIDADE DE SÃO PAULO
PARA
OBTENÇÃO DO TÍTULO
DE
MESTRE EM CIÊNCIAS

Programa: Ciência da Computação

Orientador: Prof. Dr. Daniel Macêdo Batista

Durante o desenvolvimento deste trabalho o autor recebeu auxílio da HP Brasil por meio do projeto Baile e do programa FP7/2007-2012 da Comissão Européia pelo projeto n° 257178 (CHOReOS - *Large Scale Choreographies for the Future Internet*)

São Paulo, Março de 2013

Detecção de Violações de SLA em Coreografias de Serviços Web

Este exemplar corresponde
à redação
do texto de dissertação
para realizar o depósito

Comissão Julgadora:

- Prof. Dr. Daniel Macêdo Batista (orientador) - IME-USP
- Prof. Dr. Nelson Luis Saldanha da Fonseca - IC-Unicamp
- Prof. Dr. Marco Aurélio Gerosa - IME-USP

Agradecimentos

Primeiramente eu quero agradecer à minha família. À minha mãe Saturnina, pelo grande e constante apoio apesar da distância. Ao meu pai Florentino, pela confiança que sempre deposita em mim. Ao meu irmão Fernando, porque ele também é um bom amigo. Aos meus tíos, avos, primos, obrigado a todos vocês pelo carinho e afeto que sempre me dão.

Gostaria de expressar meus sinceros agradecimentos ao meu orientador Daniel Macêdo Batista. Ele é uma ótima pessoa, excelente orientador e bom amigo. Eu agradeço ao Daniel por todas as dicas, palavras e principalmente por acreditar em mim.

Durante todo este período de mestrado, não somente aprendi e obtive conhecimento da área. Neste anos eu conhecí ótimas pessoas, as quais são de diversas cidades, países, culturas e costumes que ampliaram minha visão do mundo. Estou agradecido e fico muito contente de tê-las como boas amizades. A Valentina, a Florence, o Rafael, o Everton, a Deysi, o Enrique, o Marcelo, o Rogerio, o Lucas, o Marcos, o Fabio, o Jacques, Yanik, entre outros muitos que eu considero bons amigos.

Quero agradecer também aos meus bons amigos que viemos de Arequipa para iniciar o mestrado aquí, o Carlos e a Rosario. Ao pessoal e funcionários do IME pela amizade oferecida. Ao pessoal de Computação como o Lucas, o Fabio, o Wesley, o Strauss, o William, o Felps, o Glaucus, o Victor, entre outros. Ao pessoal do laborátorio do LIAMF, ao pessoal de Visão Computacional, ao pessoal de Matemáticas e Estatísticas. Aos meus compatriotas e bons amigos no IME como o Alvaro, o Carlos, a Rosario, o Frank, o Leandro, o Edu, a Ericka, o Juan, o Jesús Mena, a Karina, o Daniel, a Edith, o Miguel, o Jesus Mesias, o Caratos, o Mija, o Harry, o Pablito, o Geiser, a Evelyn, o Jorge, a Leissi, entre outros. Também eu gostaria de agradecer aos meus compatriotas e bons amigos da Poli-USP e da Física.

Outro grupo de excelentes amigos que gostaria de agradecer são da turma de graduação como o Fabian, o Ysaacx, o Yayo, o Carlos, o DJ, o Robert, o Ernest. Além do mais, agradeço a boa amizade de amigos que também estão estudando, ou ensinam em outros paises e que compartilham suas boas experiências, tais como o Jorge, o Omar, a Claudia, o Yayo, o Jesus Bellido, a Lisney, o Paiche, a Rosa, entre outros.

Párrafo aparte merecem o pessoal do CHOReOS e Baile. Aos professores Daniel, o Fábio, o Alfredo e o Marco, pelos conselhos e incentivos e porque sua experiência serviu no meu aprendizado. Aos membros alunos desses projetos, tais como o Felps, o Cadú, o Felipe, o Gustavo, o Eduardo, o Leonardo, o Nelson, o Daniel Cukier, e todos os demais membros dessa excelente galera.

Em especial agradeço ao CNPQ e aos projetos Baile e CHOReos pelo financiamento e incentivo a minha pesquisa. Por fim, quero terminar com uma frase que sempre levo em consideração:

> "Nunca conheci um homem tão ignorante que fosse impossível aprender algo dele".
> Galileo Galilei

Resumo

Coreografias de serviços Web representam uma forma mais escalável e flexível de compor serviços do que uma abordagem centralizada como a orquestração, e seu papel na integração e comunicação de sistemas de larga escala é vital para os objetivos da SOC (Computação Orientada a Serviços) e o Internet do Futuro. Atualmente coreografias de serviços Web possuem vários desafios de pesquisa, dos quais a qualidade de serviço (QoS) e monitoramento de coreografias de serviços Web são linhas importantes. O objetivo deste trabalho é propor e implementar um mecanismo de monitoramento não intrusivo de coreografias de serviços Web baseado em SLAs que especificam as restrições de atributos de QoS de maneira probabilística. Esta pesquisa propõe um mecanismo para: (1) definir requerimentos de QoS, (2) especificar contratos probabilísticos sobre parâmetros de QoS usando SLA e (2) realizar um monitoramento "não intrusivo" de coreografias de serviços Web para detectar violações de SLA.

Palavras-chave: Qualidade de Serviço (QoS), Monitoramento de Serviços Web, Coreografias de Serviços Web, Computação Orientada a Serviços, Acordo de Nível de Serviço (SLA), Internet do Futuro.

Abstract

Web services choreographies represent a more scalable and flexible way to compose than a centralized approach as the orchestration, and its role in the integration and communication of large-scale systems is vital for the goals of SoC (Service Oriented Computing) and Future Internet. Currently Web services choreographies have several research challenges, such as quality of service (QoS) and monitoring of Web services choreography are important research lines. The goal this work is to propose and implement a mechanism for non-intrusive monitoring of Web services choreography based on SLAs in order to define constraints of QoS attributes. This research proposes a mechanism to: (1) define QoS requirements, (2) specify contracts on probabilistic QoS parameters using SLAs and (2) carry out monitoring of Web services choreographies to detect SLA violations.

Keywords: Quality of Service (QoS), Web Service Monitoring, Web Service Choreography, Service-Oriented Computing (SoC), Service Level Agreement (SLA), Future Internet.

Sumário

Lista de Abreviaturas **vii**

Lista de Figuras **viii**

Lista de Tabelas **x**

1 Introdução **1**
1.1 Objetivos . . . 2
1.2 Considerações Preliminares . . . 2
1.3 Organização da Proposta . . . 3

2 Conceitos Básicos **4**
2.1 Serviços Web . . . 4
2.1.1 Padrões e tecnologias básicos . . . 6
2.2 Computação Orientada a Serviços . . . 7
2.2.1 Conceito de SOA . . . 8
2.2.2 Serviços Web e SOA . . . 9
2.2.3 Padrões de Desenho . . . 9
2.3 Composição de Serviços . . . 11
2.4 Internet do Futuro: A Internet das Coisas . . . 13
2.4.1 Tecnologias e Tendências . . . 15

3 Coreografias de Serviços Web **17**
3.1 Definição . . . 17
3.1.1 Padrões e Linguagens . . . 17
3.1.2 Orquestração e Coreografia de serviços . . . 18
3.1.3 Processos de Negócio . . . 19
3.2 Elementos de um Modelo de Coreografia . . . 21
3.2.1 Modelos baseados em Autômatos . . . 22
3.2.2 Modelos baseados em Redes de Petri . . . 22
3.2.3 Modelos baseados em Álgebra de Processos . . . 22
3.3 Linguagens de Coreografias de Serviços . . . 23
3.4 Ferramentas e Arcabouços . . . 24
3.5 Coreografias em BPMN . . . 24
3.5.1 BPMN . . . 24
3.5.2 Coreografia de Processos . . . 26

3.5.3 O Modelo de Interação . . . 28
3.6 Considerações Finais . . . 29

4 Cenários de Coreografias de Serviços 31
4.1 Oquestração contra Coreografias . . . 31
4.2 Cénarios . . . 31

5 QoS e Monitoramento em Coreografias de Serviços Web 36
5.1 Qualidade de Serviço . . . 36
5.1.1 Topologia de atributos e métricas de QoS . . . 39
5.2 QoS em Sistemas Orientados a Serviços . . . 39
5.3 SLAs Probabilísticos . . . 44
5.4 Trabalhos Relacionados: QoS em Coreografias de Serviços Web . . . 45
5.5 Monitoramento em Coreografias de Serviços Web . . . 46
5.6 Trabalhos Relacionados . . . 47
5.6.1 Monitoramento de Coreografias de Serviços Web Baseado em QoS . . . 47
5.6.2 Monitoramento de Serviços Usando SLAs Probabilísticos . . . 48
5.7 Considerações Finais . . . 49

6 Metodologia 50
6.1 Visão Geral . . . 50
6.2 Preliminares . . . 50
6.2.1 Modelo de QoS . . . 52
6.2.2 Modelo de Falhas . . . 54
6.3 Definição de requisitos de QoS analiticamente . . . 54
6.3.1 Definição formal de coreografias specificadas em BPMN . . . 54
6.3.2 Mapeamento de coreografias em BPMN 2.0 para GSPNs . . . 55
6.4 Simulador de Coreografias de Serviços . . . 55
6.4.1 Trabalhos Relacionados . . . 55
6.4.2 Desenvolvimento . . . 57
6.4.3 Arquitetura . . . 58
6.4.4 Definição de requerimentos de QoS . . . 59
6.5 Especificação de SLAs Probabilísticos . . . 60
6.6 Monitoramento e Detecção de Violações de SLA . . . 60
6.6.1 Detecção de violações de SLA . . . 61
6.7 Considerações Finais . . . 62

7 Resultados e Discussões 64
7.1 Cenários de Coreografia . . . 64
7.2 Definição de Requisitos de QoS Analiticamente . . . 64
7.2.1 Mapeamento . . . 64
7.2.2 Definição de pesos . . . 65
7.2.3 Simulações . . . 66
7.2.4 Análise de Resultados . . . 66
7.3 Simulador de Coreografias . . . 67

7.4 Definição de Requisitos de QoS em Coreografias 68
7.4.1 Metodologia dos Experimentos 68
7.4.2 Análise de Resultados 69
7.5 Estabelecimento de contratos de QoS 70
7.6 Monitoramento de QoS em Coreografias 71
7.7 Considerações Finais 75

8 Conclusões **76**
8.1 Considerações Finais 76
8.2 Trabalhos Futuros 77

A Especificação da Coreografia para o monitoramento no ChorSim **78**

B Configuração da plataforma para o monitoramento no ChorSim **80**

C Configuração da mmplantação para o monitoramento no ChorSim **82**

Referências Bibliográficas **84**

Lista de Abreviaturas

B2B	Negócio a Negócio (*Business to Business*)
BPEL	*Business Process Execution Language*
BPEL4Chor	*Business Process Modeling Notation for Choreography*
BPMI	*Business Process Management Initiative*
BPMN	*Business Process Modeling Notation*
CPP	*Collaboration Protocol Profile*
ebXML	*Electronic Business XML*
ESB	*Enterprise Service Bus*
GSPN	*Generalized Stochastic Petri Net*
OWL-S	*Semantic Web Ontology Language*
DAML-S	*DARPA Agent Markup Language*
OMG	*Object Management Group*
P2P	Ponto a Ponto (*peer-to-peer*)
PDF	Função Densidade de Probabilidade (*Probability Density Function*)
QoS	Qualidade de Serviço (*Quality of Service*)
REST	*Representational State Transfer*
RTT	*Round Trip Time*
SOA	Arquitetura Orientada a Serviços (*Service Oriented Architecture*)
SOC	Computação Orientada a Serviços (*Service Oriented Computing*)
SLA	Acordo de Nível de Serviço (*Service Level Agreement*)
SOAP	*Simple Object Access Protocol*
TI	Tecnologia de Informação
UDDI	*Universal Description, Discovery and Integration*
UML	*Unified Modeling Language*
W3C	*World Wide Web Consortium*
WS	Serviço Web (*Web Service*)
WSCI	*Web Service Choreography Interface*
WSDL	*Web Service Description Language*
WS-CDL	*Web Service Choreography Description Language*
WSI	*Web Services Interoperability*
WSLA	*Web Service Level Agreement*
XML	*Extensible Markup Language*

Lista de Figuras

2.1 Modelo da arquitetura de serviços Web 5
2.2 Modelo integral dos serviços Web [Pel05] 7
2.3 Evolução das arquiteturas de sistemas de software (traduzida de [Pel05]) 8
2.4 Roteiro de pesquisas em SOC e SOA [PTDL07] 9
2.5 Arquitetura genérica para um sistema de composição de serviços Web [EA06] 12
2.6 Uma nova dimensão nas tecnologias de Informação [PTDL07] 14
2.7 Tendências no Internet do Futuro [TW10] 16

3.1 A coreografia define colaborações entre participantes que interagem entre si [ADHR05] 18
3.2 Orquestração Vs. Coreografia [Pel03] 20
3.3 Coreografia como solução para modelagem de processos de negócio inter-organizacionais 20
3.4 Categorização de linguagens de coreografias de serviços Web (baseada em [Eng09]) . 23
3.5 Conjunto dos principais elementos do BPMN [DDO07] 25
3.6 Exemplo de modelo de interconexão de coreografias, oferta de investimento [KLW10]. 27
3.7 Representação de uma tarefa de coreografia [NF10] 28
3.8 Uma tarefa de coreografia simples e seu equivalente em diagrama de colaboração [NF10] 29
3.9 Uma tarefa de coreografia de dois sentidos e seu equivalente em diagrama de colaboração [NF10] 29
3.10 Exemplo de modelo de interação de coreografias, oferta de investimento [KLW10]. . 30

4.1 Cenário de *redshift* do AstroGrid: (a) orquestração, (b) coreografia [BWR09]. . . . 32
4.2 Coreografia de SLA da rede de valor de negócio [UHS11]. 34
4.3 Coreografia de serviços da aplicação de CDN [BDF+08] 35

5.1 Classificação de atributos de QoS (traduzida de [ADHR05]) 37
5.2 Tempos na invocação de serviços (traduzida de [MRLD09]) 37
5.3 Modelo multi-camada de QoS para coreografias de serviços Web (baseada em [Ros09]) 40
5.4 QoS habilitada no modelo de serviços Web de acordo com WSDL [Pan10] 41
5.5 Exemplo de SLA [KL03] 43
5.6 Definição de SLAs na camada de coreografias de serviços 44
5.7 Registro de medidas do tempo de resposta para um serviço Web (traduzida de [RBHJ08]) 45

6.1 Arquitetura do monitoramento de coreografias baseado em SLAs probabilísticos . . . 51
6.2 Elementos BPMN para modelagem de coreografias que são levados em consideração. 52

6.3 Interação de serviços a partir de interações atômicas do BPMN2. 52
6.4 Interação de serviços a partir de interações atômicas do BPMN2. 53
6.5 Atributos de QoS em uma interação com um serviço Web. 53
6.6 Mapeamento de eventos e *gateways* para módulos de redes de Petri 56
6.7 Mapeamento de dois tipos de tarefas de coreografia para módulos de redes de Petri com suporte de QoS 56
6.8 Atributos de QoS calculados em um evento dado. (1) Recebendo requisições de um cliente ou serviço. (2) enviando requisições para um outro serviço. (3) recebendo resposta de um outro serviço (dependência). (4) enviando resposta para um cliente ou serviço solicitador. 58
6.9 Arquitetura do simulador de coreografias. 59

7.1 Coreografia de serviços da aplicação de CDN [BDF+08] 65
7.2 GSPN obtido da coreografia do cenário de exemplo 65
7.3 Tempos de reposta médio, máximo e minimo de acordo ao número de requisições concorrentes. 67
7.4 Modelo de falhas que mostra a largura de banda efetiva devido à degradação da largura de banda referencial em um período de 100 segundos. 68
7.5 Tempo médio de resposta total da coreografia em função do tamanho de resposta do serviço WS1 com larguras de banda de 1Mbps até 16Mbps (intervalos de confiança não são visíveis por terem ficado muito pequenos). 69
7.6 Tempo médio de resposta total da coreografia em função do tamanho de resposta do serviço WS1 segundo o modelo de falha. A largura de banda varia de 1Mbps até 16Mbps 70
7.7 Distribuição de probabilidade empírica (ECDF) do contrato do serviço WS_1 com base nos tempos de resposta 71
7.8 Monitoramento e detecção de violações de SLA para o cenário 1. 73
7.9 Monitoramento e detecção de violações de SLA para o cenário 2 73
7.10 Monitoramento e detecção de violações de SLA para o cenário 3 74
7.11 Monitoramento e detecção de violações de SLA para o cenário 4 74

Lista de Tabelas

5.1 QoS no modelo WSDL estendido 42
5.2 Quantis de tempos de resposta (obtida de [RBHJ08]) 45

6.1 Modelo de QoS e de falhas 58

7.1 Descrição das posições da GSPN resultante 65
7.2 Descrição de transições da GSPN resultante 66
7.3 Configuração de pesos nos Cenários 1 e 2 66
7.4 Resultados (em %) 67
7.5 Configuração dos cenários de simulação 68
7.6 Configuração de valores dos atributos de QoS nas requisições 69
7.7 Configuração de valores dos atributos de QoS nas respostas 69
7.8 Configuração das taxas de degradação dos serviços para obter o contrato do tempo de resposta para o serviço composto WS_1 70
7.9 Quantis do contrato probabilístico dos tempos de resposta de WS_1. 71
7.10 Definição de valores no configuração para o monitoramento. 72
7.11 Estabelecimento das taxas de degradação de tempo de processamento dos serviços para os cenários. 72

Capítulo 1

Introdução

O modo de desenvolver aplicações e sistemas complexos tem evoluído com o passar do tempo até convergir, atualmente, para arquiteturas de software e modelos de computação orientados a serviços, o que é chamado de Computação Orientada a Serviços (SOC). SOC é um paradigma que utiliza serviços como base para construir e suportar um desenvolvimento rápido, de baixo custo, de baixo acoplamento e de fácil composição de aplicações heterogêneas distribuídas [PTDL08]. Essas composições podem ser realizadas principalmente de duas formas: orquestração e coreografias, das quais a coreografia de serviços é mais colaborativa e especifica a composição desde uma perspectiva global. Dada a natureza distribuída e colaborativa de uma coreografia de serviços, ela é indicada como a melhor alternativa de integração de sistemas complexos e heterogêneos em larga escala [VGF10].

Atualmente existem duas principais abordagens para compor serviços, a orquestração e a coreografia. A orquestração de serviços é uma composição centralizada, já que uma entidade denominada *orquestrador* é responsável por coordenar a comunicação dos serviços participantes. Por outro lado, a coreografia de serviços é uma composição descentralizada já que é uma descrição de interações ponto a ponto entre os serviços participantes, ou seja, nesse modelo, não há a figura de um controlador central [BWR09].

Devido ao número crescente de dispositivos móveis que se conectam à Internet, uma abordagem orientada a serviços centralizada como a orquestração pode não ser escalável em termos de largura de banda para lidar com o número cada vez mais crescente de dispositivos e serviços. Nesse cenário, uma abordagem descentralizada, como a coreografia, pode se tornar a mais adequada para as características da Internet do Futuro [Stu12].

Por outro lado, um fator chave para a realização de comportamento adaptativo para sistemas orientados a serviços, e especificamente em composição de serviços, é a disponibilidade e suporte de informação de QoS (Qualidade de Serviço) [Ros09]. A característica distribuída das coreografias de serviços Web torna necessário que haja adaptação, a fim de que os requisitos de QoS das aplicações sejam atendidos. Como consequência, o monitoramento de QoS torna-se importante, porque observa, coleta e reporta informações de QoS em tempo de execução e durante a evolução do sistema baseado em serviços [JDM10]. A partir do monitoramento é possível alavancar a ligação dinâmica, seleção, composição, adaptação, autocura (*self-healing*), otimização, entre outros, de serviços Web baseada na QoS [Ros09].

A forma mais trivial de representar os requisitos de QoS é através de contratos rígidos, em que as métricas devem ser respeitadas à risca em 100% dos casos (por exemplo, atraso $< 10ms$). Entretanto, esses contratos não representam o comportamento dinâmico dos atributos de QoS dos serviços Web. Porém, uma melhor forma de representar os requisitos de QoS é através de contratos não rígidos e baseados em probabilidade. Neste caso os requisitos estipulam distribuições de probabilidade relacionadas com os valores dos requisitos de QoS (por exemplo, atraso $< 10ms$ em 95% das vezes em que a métrica for monitorada). Além disso, assim como ocorre em outros sistemas em redes [BCdFZ10], é importante que o monitoramento das métricas de QoS seja feito de forma não intrusiva e que retorne valores bem próximos dos valores reais.

No entanto, atualmente, implementar e executar uma coreografia de serviço real é uma tarefa complexa já que a tecnologia para suportar esse paradigma de composição de serviços está imatura, especialmente pela falta de motores de execução cientes de coreografia [KELL10]. Assim, os mecanismos para medir parâmetros de QoS, e estabelecer requisitos de QoS não estão bem desenvolvidos para coreografias.

Esta pesquisa propõe um monitoramento de coreografias de serviços Web baseado em restrições de atributos de QoS de maneira probabilística com o objetivo de detectar violações de SLAs entre os participantes da coreografia. O monitoramento será realizado de maneira não intrusiva, e é desenvolvido acima do nosso simulador para realizar *enactment* de coreografias de serviços. Tal simulador possui suporte de QoS, principalmente de de tempo de processamento, tempo de resposta e atributos de rede. Desse modo, o monitor é responsável pela coleta e medição dos atributos de QoS dos serviços individuais, e de agregá-los de acordo à composição e finalmente verificar a existência de violação de alguma restrição de QoS segundo como está definido no contrato.

1.1 Objetivos

O objetivo principal deste trabalho é :

- Detectar violações de SLAs baseadas em restrições probabilísticas de QoS em coreografias de serviços Web.

Os objetivos secundários deste trabalho são:

- Propor mecanismos para definir requisitos de QoS em coreografias de serviços Web.
- Propor uma técnica de monitoramento de coreografias de serviços Web com restrições de QoS definidas probabilisticamente.
- Desenvolver um simulador de coreografias de serviços Web com suporte de QoS.
- Realizar avaliações do mecanismo de monitoramento de coreografias com simulações e medições.

1.2 Considerações Preliminares

Para o cumprimento dos objetivos deste trabalho, consideram-se várias etapas:

1. Desenvolvimento de um simulador de coreografias de serviços Web com suporte de QoS.
2. Definição de requisitos de QoS e estabelecimento de contratos entre os serviços participantes que interagem em uma coreografia.
3. Realização do monitoramento da coreografia, estimando as distribuições de probabilidade dos parâmetros de QoS.

Na etapa (1) construiu-se um simulador de coreografias por conta da falta de implementações para rodar coreografias com suporte de QoS. Na etapa (2) se definem requisitos de QoS baseados em avaliações de desempenho de coreografias usando modelos de falhas probabilísticos de atributos de QoS. Com base nesses requisitos de QoS se estebelecem os contratos probabilísticos. Utilizaram-se duas abordagens de modo a definir requisitos de QoS:

- **Analítica**: Usando Redes de Petri Estocásticas Generalizadas (GSPN) como representação intermediária da especificação de uma coreografia e adicionando características de QoS.
- **Simulações**: Por meio de avaliações de desempenho usando o simulador de coreografias.

Na etapa (3), o monitoramento utiliza os quantis das distribuições de probabilidade empírica (de acordo com os valores medidos dos parâmetros de QoS), e os compara com os quantis definidos nos SLAs para verificar alguma violação do contrato. Os parâmetros de QoS que serão utilizados são o tempo de resposta, o tempo de processamento, o atraso de rede e a largura de banda efetiva das interações entre os serviços.A reação (punição, adaptação, reconfiguração, entre outros) ante a detecção de alguma violação de um SLA não está no escopo desta pesquisa.

Esta pesquisa foi desenvolvida no contexto e apoio de dois projetos:

- O projeto CHOReOS [1] - Coreografias de larga escala para o Internet do futuro.
- O projeto Baile [2] - Habilitando coreografias escaláveis de serviços na núvem.

1.3 Organização da Proposta

O restante deste texto está organizado da seguinte forma. No Capítulo 2 são apresentados os aspectos e conceitos básicos, tais como serviços Web, SOA, composição de serviços e Internet do futuro. No Capítulo 3 são descritos os conceitos de coreografia de serviços Web. Em seguida, o Capítulo 4 apresenta alguns cenários onde podem ser aplicadas as coreografias de serviços. No Capítulo 5 são descritos os fundamentos de QoS, SLA e monitoramento em coreografias de serviços, além de serem resumidos os trabalhos relacionados a esta pesquisa. Depois, no Capítulo 6 se apresentam as abordagens, técnicas e algoritmos para atingir os objetivos desta pesquisa. O Capítulo 7 descreve os experimentos realizados e se apresentam as análises dos resultados. Finalmente, o Capítulo 8 apresenta as conclusões, considerações finais e os trabalhos futuros.

[1] Projeto CHOReOS: http://www.choreos.eu
[2] Projeto Baile: http://ccsl.ime.usp.br/baile/

Capítulo 2

Conceitos Básicos

2.1 Serviços Web

Os serviços Web foram propostos para facilitar a comunicação entre componentes de arquiteturas heterogêneas, oferecendo uma visão destas arquiteturas baseada em serviços e totalmente compatível com a Internet. O surgimento dos serviços Web, e da SOA, implicaram o estabelecimento de novos mecanismos de B2B, B2C e B2E. A organização responsável pela definição dessas normativas ou padrões é o WS-I [WSI05](*Web Services Interoperability Organization*). Esta organização possibilita que os sistemas desenvolvidos em diferentes plataformas e diferentes linguagens de programação possam interagir.

O modelo de serviços Web acrescenta o desenvolvimento de aplicações distribuídas. A arquitetura de serviços Web permite que os serviços sejam descritos de forma dinâmica, publicados, descobertos e invocados em um ambiente de computação distribuída, utilizando os padrões WSDL[1], UDDI[2], SOAP[3] e XML, respectivamente. Segundo [New02], os serviços Web representam o próximo passo na evolução da tecnologia orientada a objetos e representam uma revolução, afastando-se da tradicional arquitetura cliente-servidor para as novas arquiteturas ponto a ponto.

Os serviços Web estão baseados em um conjunto de normativas (WSDL, UDDI, XML e SOAP) que permite aos programadores implementarem aplicações distribuídas. Assim, os serviços Web são aplicações auto-contidas e modulares que podem ser:

- Descritas por uma linguagem de descrição de serviços, como a linguagem WSDL.
- Publicadas, incluindo as descrições e as políticas de uso em um registro conhecido, utilizando o método de registro UDDI.
- Descobertas, também usando o padrão UDDI, para enviar requisições para o registro e receber os detalhes necessários para a localização e ligação (*binding*) do serviço que atenda os parâmetros de busca.
- Associadas, quando as informações contidas na descrição do serviço são utilizadas para criar uma instância do serviço disponível.
- Invocadas sobre a rede, utilizando as informações contidas nos detalhes da descrição do serviço, em um documento WSDL. Durante a invocação se utiliza o protocolo SOAP.
- Composta com outros serviços para integrar novas aplicações e serviços, o que é a base da SOA.

A Figura 2.1 mostra e ilustra os componentes que estão envolvidos no modelo de uma arquitetura de serviços Web. A seguir a descrição desses elementos [TPM06]:

[1]WSDL : http://www.w3.org/TR/wsdl
[2]UDDI: http://uddi.xml.org/
[3] SOAP:www.w3.org/TR/soap/

- **Serviço**: A "aplicação" publicada para ser utilizada pelos solicitantes que cumpram os requisitos especificados pelo provedor de serviços. Obviamente, a execução é realizada em uma plataforma acessível na rede.
- **Provedor do serviço**: Do ponto de vista comercial, é quem fornece o serviço. Do ponto de vista da arquitetura, é a plataforma que fornece o serviço.
- **Registro de serviços** : É um repositório de descrições, em que os provedores publicam seus serviços e as formas de acessá-los. Também permite aos solicitantes realizarem diferentes tipos de buscas.
- **Solicitante do serviço**: Do ponto de vista comercial, é a empresa que precisa de um determinado serviço. Do ponto de vista da arquitetura, é o aplicativo cliente que invoca um serviço de busca.

Figura 2.1: *Modelo da arquitetura de serviços Web*

As operações que podem ser realizadas com os serviços Web são:

- **Publicar/Cancelar**: Um provedor de serviços pode publicar um determinado serviço comercial (*e-business*) em um ou mais registros de serviços e cancelar essa publicação em qualquer momento.
- **Procurar**: Os solicitantes dos serviços (clientes) podem interagir com um ou mais registros para encontrar um conjunto de serviços que solucione seus problemas.
- **Ligação (*binding*)**: Os solicitantes negociam com os provedores de serviços a forma de acesso e invocação dos seus serviços comerciais.

Atualmente, as aplicações que estão sendo desenvolvidas têm a capacidade de procurar e selecionar serviços de forma dinâmica em tempo real, com base em parâmetros como custos, qualidade ou disponibilidade. Isto é uma grande vantagem na hora de utilizar sistemas baseados em serviços Web, já que o sistema automaticamente escolhe o serviço que melhor se adapte às suas necessidades [Wan04].

Entre as razões pelas quais os serviços Web possuem um papel importante na implementação de sistemas distribuídos estão [TPM06]:

- **Interoperabilidade:** Qualquer serviço Web pode interagir com qualquer outro serviço. O protocolo padrão SOAP permite que qualquer serviço possa ser oferecido ou utilizado independentemente da linguagem ou ambiente em que é desenvolvido.

- **Onipresença:** Os serviços Web se comunicam usando HTTP e XML. Qualquer dispositivo que trabalha com estas tecnologias pode ser o cliente e acessar os serviços Web. Por exemplo, uma máquina de venda de refrigerantes pode se comunicar com o serviço Web de um provedor local e solicitar uma ordem de fornecimento através de uma rede de acesso sem fio.

- **Barreira mínima de participação:** Os conceitos por trás dos serviços Web são fáceis de compreender e existe uma gama de ferramentas de desenvolvimento , como os oferecidos pela IBM, *Sun Microsystems*, *Apache*, *Systinet*, entre outras. Todos eles permitem que os desenvolvedores criem e implementem rapidamente serviços Web.

- **Suporte da indústria:** As principais empresas de TI suportam os serviços Web baseados em SOAP, e serviços Web baseados na arquitetura REST [4], conhecidos como serviços Web *RESTFul* [PML09], e tecnologias derivadas dos serviços Web.

2.1.1 Padrões e tecnologias básicos

A infra-estrutura mínima exigida pelos serviços Web pode ser definida em termos de [Pel05]:

- O que vai na "rede": Formatos e protocolos de comunicação.

- O que descreve e vai na rede: Linguagens de descrição de serviços.

- O que permite encontrar e armazenar essas descrições: Descoberta de serviços.

Os serviços Web baseiam o seu desempenho global e as características nas seguintes normativas e protocolos [ACKM04]:

- **XML** (*eXtensible Markup Language*) [W3C98], publicado em 1998 e tem revolucionado a maneira de estruturar, descrever e trocar informação. Independentemente das muitas maneiras atuais de usar XML, todos as tecnologias de serviços Web são baseados em XML.É a padrão central desta arquitetura, sobre a qual o resto de padrões se apoiam. O desenho de XML deriva de duas fontes principais: SGML (*Standard Generalized Markup Language*) [ISO86] e HTML(*HyperText Markup Language*) [ISO00].

- **UDDI** (*Universal Description, Discovery and Integration*) [UDD05], é um diretório que contém um registro/repositório de descrições de serviços Web. Este padrão permite que as empresas se registrem em um tipo de diretório da Internet, que as ajuda a anunciar seus serviços, de modo que outras empresas possam localizar os seus serviços e realizar transações na web. O processo de registro e consulta se realizam usando os mecanismos baseados em XML e HTTP(S). Portanto, a especificação UDDI tem dois objetivos principais, primeiro, servir de suporte aos desenvolvedores para encontrar informações sobre os serviços Web e poder construir os clientes; e por outro lado, facilitando a ligação dinâmica de serviços Web, permitindo consultar referências e acessar serviços de interesse.

- **SOAP** (*Simple Object Access Protocol*) [HMG+03], é um padrão do W3C[5], que define um protocolo que da suporte para a interação (dados + funcionalidade) entre aplicações em ambientes heterogêneos distribuídos, é interoperável (independente da plataforma, linguagem de programação, independente do hardware e protocolos). SOAP define a forma de organizar informações XML de uma forma estruturada e tipada para ser trocada entre diferentes sistemas. O protocolo SOAP simplifica o acesso a objetos, permitindo que as aplicações possam invocar métodos de objetos ou funções que residem em sistemas remotos.

[4] REST: *Representational State Transfer*
[5] W3C: World Wide Web Consortium, www.w3.org/

- **WSDL**(Web Service Description Language) [CCMW01], criado originalmente pela IBM, Microsoft e Ariba. Tem um papel e um propósito semelhante a IDL (*Interface Definition Language*) das plataformas de *middleware*. Um arquivo WSDL é um documento XML que descreve os serviços Web, particularmente suas interfaces. Uma característica que o diferencia do IDL, é que o WSDL deve definir os mecanismos de acesso (protocolos) para os serviços web. Outra característica distintiva é a necessidade de definir (na especificação) a localização de um serviço (*endpoints*). A separação das interfaces e os enlaces de protocolo, e a necessidade para incluir as informações de localização permite a definição de especificações modulares. WSDL permite definir interfaces de forma mais complexa e expressiva, permitindo definições de interações assíncronas e de diferentes paradigmas de interação, e a possibilidade de misturar ou ou agrupar operações.

Na Figura 2.2 se mostra o diagrama de arquitetura dos sistemas baseados em serviços Web, onde se representam os padrões e tecnologias descritas.

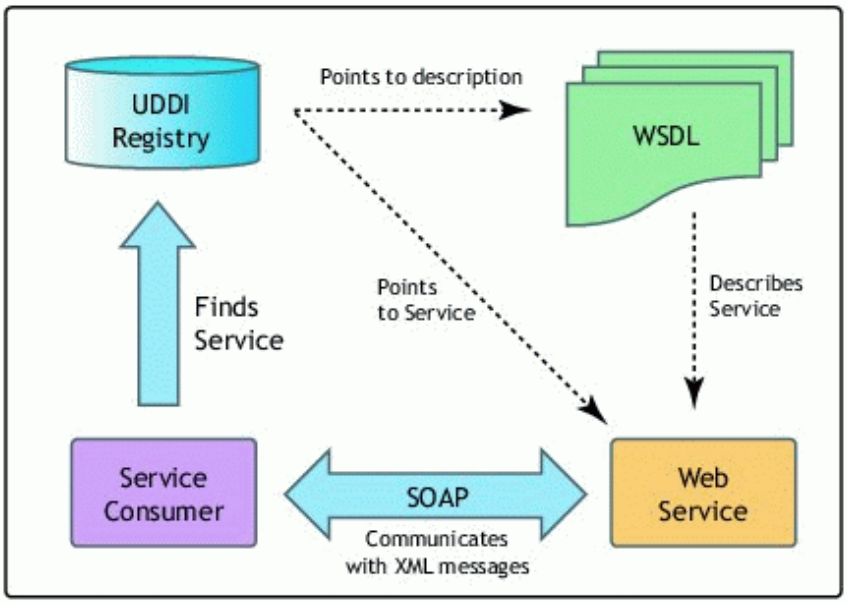

Figura 2.2: *Modelo integral dos serviços Web [Pel05]*

2.2 Computação Orientada a Serviços

Integrar produtos de diferentes provedores e de diferentes tecnologias é uma tarefa difícil, por isso cada vez mais as aplicações estão orientadas a serviços. Isto pode ser visto na Figura 2.3, na qual se apresenta a evolução das arquiteturas de software, paradigmas de programação, metodologias de desenvolvimento e também os modelos de computação tanto no hardware quanto nas redes, convergindo para a computação orientada a serviços (*Service Oriented Computing* - SOC).

O paradigma da SOC utiliza os serviços como blocos de construção para suportar o desenvolvimento de maneira rápida, interoperável, evolutiva, de baixo custo e de fácil composição de aplicações distribuídas. Serviços são entidades computacionais autônomos que que podem ser utilizados independentemente da plataforma. Os serviços podem ser descritos, publicados, descobertos, e dinamicamente montados para o desenvolvimento massivo de sistemas distribuídos, interoperáveis e evoluíveis. Os serviços desempenham funções que podem variar de responder requisições simples para a execução de processos de negócios sofisticados quem exigem relações P2P entre várias camadas de consumidores e provedores de serviços.

Qualquer troço de código e qualquer componente de um aplicativo implantado em um sistema pode ser reutilizado e transformado em um serviço disponibilizado na rede. Serviços refletem uma

Figura 2.3: *Evolução das arquiteturas de sistemas de software (traduzida de [Pel05])*

abordagem "orientada a serviços"na programação, com base na idéia de composição de aplicativos pela descoberta e invocação de serviços disponíveis na rede ao invés de construir novos aplicativos ou invocar aplicativos específicos. Os serviços são mais frequentemente construídos de maneira independente do contexto em que que são utilizados. Isso significa que o provedor e consumidor de serviços são baixamente acoplados.

A abordagem "orientada a serviços"é independente de linguagens de programação específicas ou sistemas operacionais. Isso permite que as organizações exponham suas funcionalidades essenciais por meio da Internet ou de uma variedade de redes, por exemplo, cabo, UMTS, XDSL, Bluetooth, etc usando linguagens (baseadas em XML) e protocolos padrão, e implementando uma interface auto-descriptiva. Atualmente, os serviços Web são a tecnologia mais promisoria baseada no conceito de SOC. A promessa visionária de SOC consiste em construir facilmente componentes de aplicação dentro de uma rede de serviços de baixo acoplamento, a fim de que se possa criar processos de negócios dinâmicos e aplicações ágeis que cubram diversas organizações e plataformas de computação [PTDL08].

2.2.1 Conceito de SOA

A chave para concretizar a visão da SOC é a arquitetura orientada a serviços (SOA). SOA é uma maneira lógica de desenhar sistemas de software para fornecer serviços para aplicações de usuário final ou outros serviços distribuídos em uma rede, por meio de interfaces publicadas e detectáveis (isto é, que possam ser descobertas) . Uma aplicação SOA bem construída, baseada em padrões, pode fortalecer uma organização com uma infra-estrutura flexível e com um ambiente de processamento. Isto por causa de um provisionamento independente,funções de aplicações reutilizáveis como serviços, e provendo uma base robusta para alavancar esses serviços [PTDL07].

Uma Arquitetura Orientada a a Serviços (SOA) é uma abordagem emergente que orienta os requerimentos de baixo acoplamento, baseado em padrões e uma computação distribuída independente de protocolos [PH07]. Na Figura 2.4 mostra um roteiro de pesquisas em SOA segundo [PTDL07]), que separa a funcionalidade em três planos. Serviços básicos na base da pirâmide, composição de serviços no meio e gerenciamento de serviços com monitoramento no topo da pirâmide. Esta divisão lógica está baseada na necessidade de separar:

- Capacidades de serviços básicos fornecidos por uma infraestrutura de *middleware*, e uma SOA convencional de funcionalidades de serviços avançada necessária para dinamicamente compor serviços.
- Serviços de negócio de serviços focados em sistemas.
- Composição de serviços de gerenciamento de serviços.

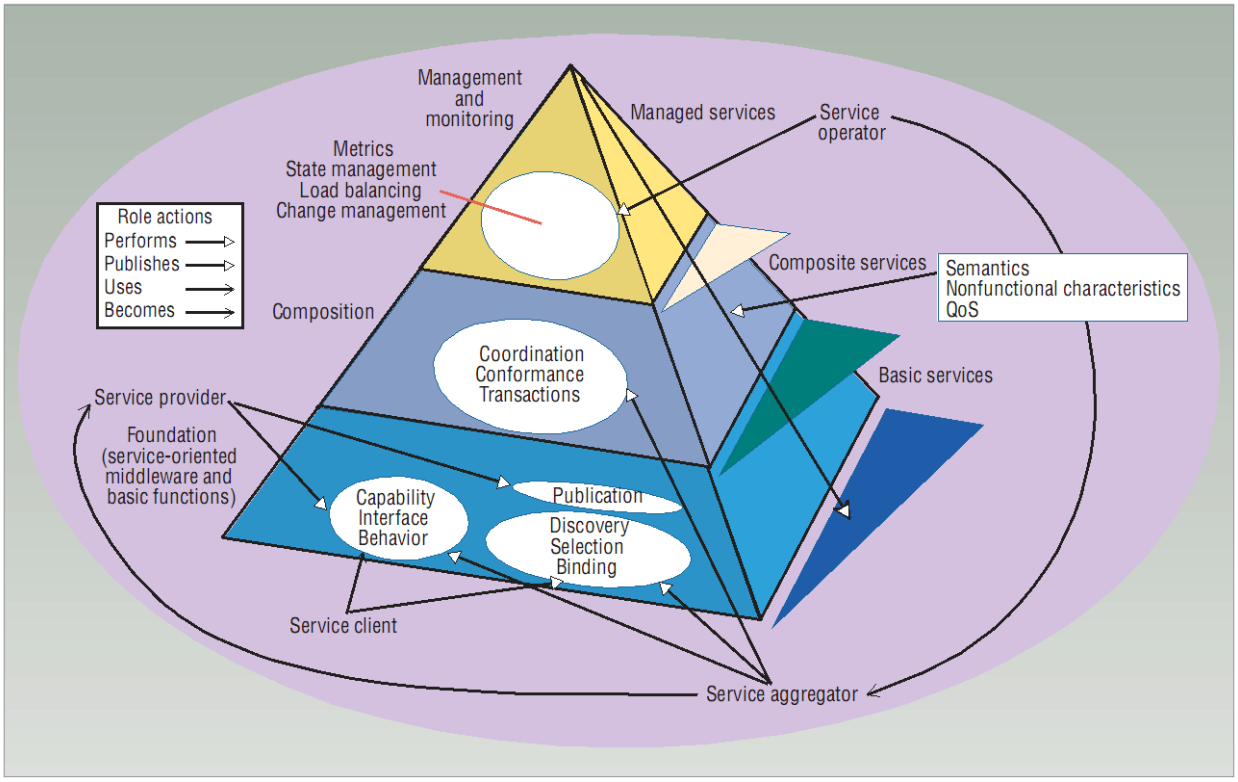

Figura 2.4: *Roteiro de pesquisas em SOC e SOA [PTDL07]*

2.2.2 Serviços Web e SOA

Este ponto é justificado porque os serviços Web podem se tornar uma parte vital de uma arquitetura orientada a serviços. Dentro de um processo de negócio pode ser necessário utilizar funcionalidades de diferentes sistemas e de diferentes localizações, para realizar essas funções se utilizam os serviços Web, estes também têm de ser identificados durante o processo de análise de uma arquitetura orientada a serviços. Cada serviço deve ser bem definido por uma interface (WSDL) para que possa ser publicado, localizado e invocado. Dependendo do processo de negócio, o serviço pode ser publicado para que outras empresas podem usá-lo, ou internamente, para uso em processos de negócio internos de uma empresa particular. Os serviços Web são uma tecnologia altamente adaptável às necessidades de implementação de uma arquitetura orientada a serviços. Essencialmente, os serviços Web são a implementação de uma especificação bem definida de uma funcionalidade, isto é, são aplicações modulares que fornecem uma lógica de processos de negócio como um serviço que pode ser publicado, localizado e invocado através da Internet [IBM06]. Baseado no padrão XML [W3C98], os serviços Web podem ser desenvolvidos utilizando qualquer linguagem de programação, qualquer protocolo e qualquer plataforma. Os Serviços Web podem ser localizados e utilizados a qualquer momento em qualquer localização e usando qualquer protocolo e plataforma. Mas é importante notar que os serviços Web não são a única tecnologia usada para implementar um SOA. Existem muitos exemplos de organizações que utilizam com sucesso SOA, no qual se utilizam (além dos serviços Web) outras tecnologias de troca de mensagens e de funções de acesso remoto, usando o padrão XML, como o protocolo XML-RPC [Win99], que funciona exatamente como o protocolo RPC (Remote Procedure Call), através de um túnel de HTTP.

2.2.3 Padrões de Desenho

Para a construção de aplicações orientadas a serviços usando SOA, também foi desenvolvido uma série de padrões de desenho de software [GHJV94]. Com o uso de padrões, que podem ser usados com qualquer metodologia, o SOA resultante irá conter um maior nível de abstração, garantindo

consistência e melhor desempenho. Os padrões de desenho para construir uma arquitetura orientada a serviços podem ser divididos em cinco categorias [Mon03]:

1. **Aprendizagem**: Utilizado para compreender o ambiente dos serviços Web. Dentro desta categoria encontramos:
   - Arquitetura Orientada a Serviços: O padrão que forma a arquitetura dos serviços Web.
   - *Architecture Adapter*: Ele pode ser visto como um modelo genérico que facilita a comunicação entre arquiteturas.
   - *Service Directoy*: Esse padrão facilita a transparência na localização de serviços, permitindo fazer interfaces robustas para encontrar o serviço que eles realmente querem.
2. **Adaptação**: Estes padrões básicos são chamados de básicos para conhecer o funcionamento do entorno dos serviços web. Nesta categoria encontramos:
   - *Bussines Object*: Um business object engloba um conceito de negócio no mundo real, como um cliente, uma empresa ou produto, e o que pretende este padrão é transferir o conceito de objeto de negócios dentro do paradigma de Web Services.
   - *Business Process*: Este padrão é utilizado no tratamento de processos de negócio. Atualmente, existem duas especificações principais:
     - Business Process Execution Language (BPEL), proposto pela BEA Systems, IBM e Microsoft.
     - Business Process Modeling Notation (BPMN), proposto por outras empresas que não estão no primeiro grupo, tais como WebMethods, SeeBeyond, etc.
   - *Bussines Object Collection:* Este padrão pode fazer composições de processos de negócios.
   - *Asynchronous Business Process*: Este padrão é a evolução do padrão Business Process.
3. **Alterações**: Embora os serviços Web permitam chamadas assíncronas, a implementação do serviço pode ser baseada no passo de mensagens, também são importantes serviços baseados em eventos, esses padrões são baseados em padrões tradicionais, como o Observer ou o padrão Publish / Subscribe. Nesta categoria encontramos:
   - *Event/Monitor*: Este é um padrão para criar formas eficazes de integrar aplicações sem a interferência de outros componentes. O cenário mais comum de este padrão é utilizado para aplicações de EAI (Enterprise Application Integration) [GZ10].
   - *Observer Services*: Este padrão representa a forma mais natural para detectar mudanças e agir devidamente.
   - *Publish / Subscribe Services*: A evolução do padrão Observer, enquanto o padrão Observer é baseado no registro, este padrão é baseado nas notificações, o que permite diferentes serviços possam enviar a mesma notificação.
4. **Redefinição**: Esses padrões permitem acesso ao comportamento de um serviço que é implementado em uma linguagem. Eles ajudam a compreender o ambiente do serviços Web e para moldar o ambiente para atender às nossas necessidades. Nesta categoria encontramos:
   - *Physical Tires*: Este padrão ajuda a estruturar melhor a lógica de negócio dos serviços Web, e pode até mesmo ser usado para controlar o fluxo de negociações pode ser obtido com o padrão *publish/subscribe.*
   - *Connector*: Este padrão é normalmente utilizado com o anterior para resolver quaisquer problemas que surjam na subscrição.

- *Faux Implementation*: É uma alternativa para resolver os problemas que surgem na utilização de eventos em serviços Web. É simplesmente uma socket aberto, que recebe as conexões e fornece as respostas para os diferentes eventos.

5. **Flexibilidade**: Para criar serviços mais flexíveis e otimizados. Nesta categoria estão:

   - *Service Factory*: Um dos principais padrões e permite a seleção de serviços e fornece flexibilidade na instanciação dos componentes que criam os serviços Web. Esse padrão também é usado tipicamente com o padrão de Service Cache para proporcionar maior flexibilidade na manutenção de aplicações que utilizam os serviços Web, oferecendo um maior ROI às aplicações.
   - *Data Transfer Object*: Esta norma fornece desempenho, permitindo coletar dados e enviar múltiplas em uma única chamada, reduzindo o número de conexões que o cliente tem que fazer para o servidor.
   - *Data Transfer Collection*: Esta é uma extensão da primeira, dado que o padrão Data Transfer Object pode ser aplicado a uma coleção de objetos de negócio. Este objeto pode retornar um conjunto de atributos comuns de uma coleção de objetos.
   - *Parcial Population*: Este padrão permite aos usuários selecionar somente os dados que são necessários para as suas necessidades e só recuperar do servidor aquilo que é necessário. Este padrão além de desempenho também oferece mais largura de banda na rede.

Alguns padrões usam outros padrões, por exemplo, o padrão *Business Process* utiliza o *Business Object* e o *Business Object Collection*. E a *Service-Oriented Architecture* faz uso *Service Directory* e *Architecture Adapter*.

2.3 Composição de Serviços

A composição de serviços Web em um novo serviço é um grande desafio. Há duas formas principais de compor serviços Web: orquestrações e coreografias. O Capítulo 3 entrará em detalhes sobre cada uma dessas duas formas. Resumidamente, composições baseadas em orquestrações possuem uma entidade central que controla a chamada a cada um dos serviços da composição. Composições baseadas em coreografias são colaborativas e não possuem nenhuma entidade central como existe nas orquestrações.

A Figura 2.5 apresenta uma arquitetura genérica para descrever o ciclo de vida em uma composição de serviços Web. Para iniciar um processo de composição, deve-se descrever e publicar um serviço (Etapa 1 da Figura 2.5). Em seguida deve haver uma solicitação para iniciar o processo de composição. O processo de definição e composição dos serviços (Etapa 2 da Figura 2.5) pode invocar a tradução de uma linguagem de design para uma linguagem mais formal, que é usada pelo gerador do processo de composição (Etapa 3 da Figura 2.5). Já que alguns serviços podem ter funcionalidades similares, pode ser gerado mais de um modelo de serviço de composição que atenda às requisições. Neste caso, os serviços compostos são avaliados (Etapa 4 da Figura 2.5), e finalmente o serviço composto escolhido é executado e monitorado (Etapa 5 da Figura 2.5).

A maioria das ferramentas que apoiam o projeto e definição da composição de serviços fazem uma distinção entre linguagens de especificação de serviços internas e externas. As linguagens externas (também chamadas de linguagens de design) permitem representar, geralmente em forma gráfica, a composição de serviços para que ela possa ser facilmente compreendida pelos interessados. Por outro lado, as linguagens internas (linguagens de composição de serviços) são linguagens geralmente mais precisas e formais, que são usados para gerar o processo de composição.

A lista a seguir apresenta algumas linguagens externas:

- **WS-BPEL**: É uma linguagem com uma sintaxe baseada em XML que suporta a especificação de processos de negócio, que por sua vez utilizam operações que envolvem um ou vários

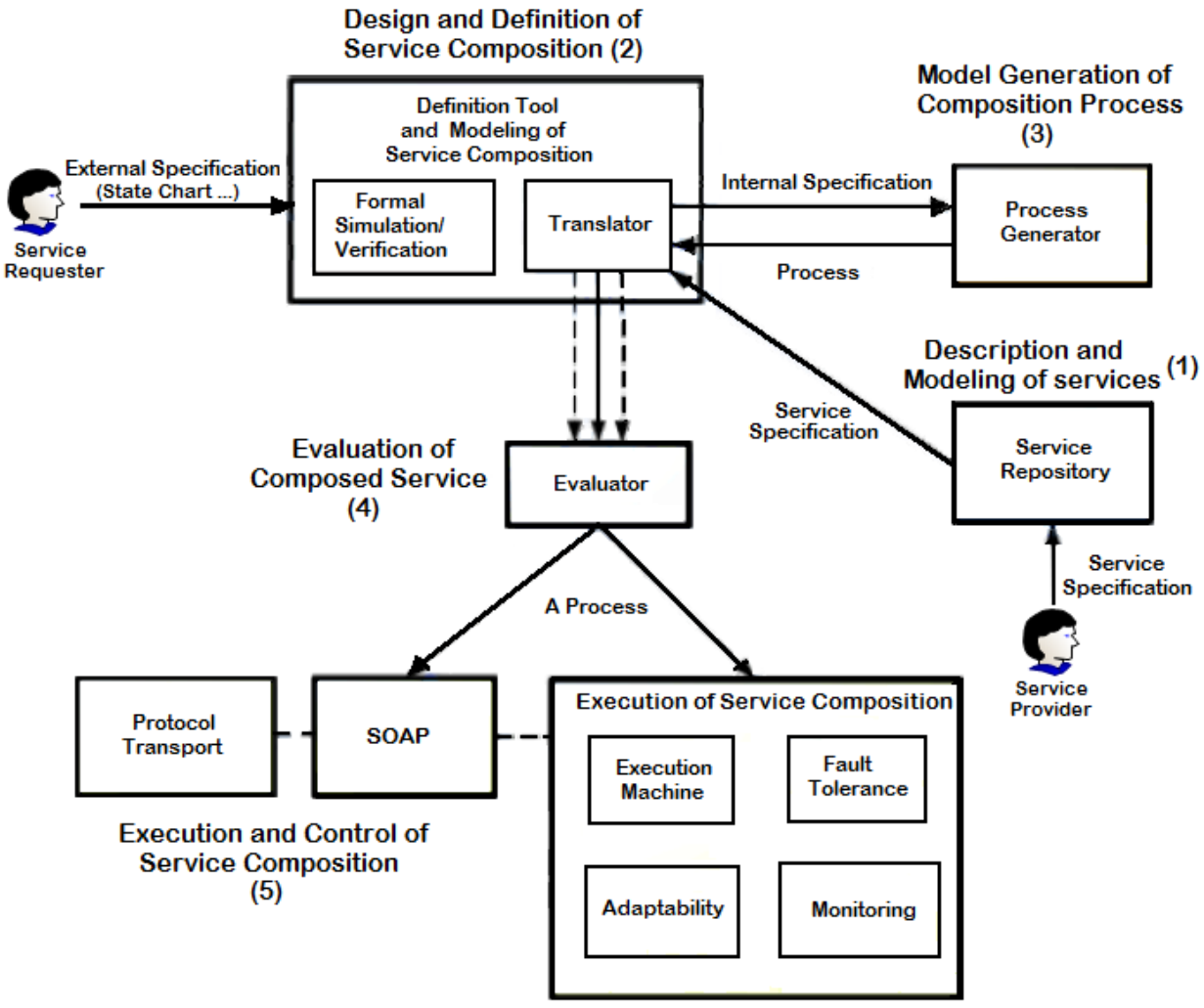

Figura 2.5: *Arquitetura genérica para um sistema de composição de serviços Web [EA06]*

serviços Web. BPEL (nome curto do WS-BPEL) suporta a descrição de dois tipos de processos: abstrato e executável. A linguagem utiliza os conceitos desenvolvidos na área de gestão de fluxos de trabalho (*workflows*) e é relativamente expressiva, em comparação a algumas linguagens suportadas por sistemas de fluxos de trabalho existentes e padrões relacionados.

- **ebXML BPSS**: *Electronic Business XML*, é um conjunto de padrões com o objetivo de fornecer uma plataforma de implementação para a colaboração entre negócios. ebXML adota uma proposta baseada na coreografia para a composição de serviços. Especificamente, uma colaboração de negócios é descrita como um conjunto de Perfis de Protocolos de Colaboração (CPP).

- **BPML**[6]: O BPMi (*Business Process Management Initiative*) é um consórcio que tem como objetivo contribuir no desenvolvimento de padrões de descrição de processos. Esse consórcio tem publicado uma especificação para uma linguagem de descrição de processos chamada de BPML (*Business Process Modeling Language*), que é similar em algumas formas a WS-BPEL, e está voltada para a descrição de orquestrações de serviços Web. BPML se beneficia de um padrão prévio chamado de WSCI desenvolvido pelos interessados em BPMi. Embora descontinuada, a WSCI integra alguns dos modelos encontrados no BPML e BPEL (por exemplo, a sequência, a execução paralela e as primitivas para enviar/receber) e é usada também para descrever coreografias de serviços Web.

[6]BPML: Atualmente é BPMN

As linguagens externas possuem limitações para o projeto e análise de uma composição de serviços, por exemplo, para permitir técnicas de simulação, e para garantir a verificação de propriedades como segurança, gerenciamento de recursos e satisfação das restrições de negócio. Por causa disso, alguns grupos de pesquisa têm proposto algumas linguagens formais de composição de serviços. A seguir, descrevem-se algumas das linguagens formais para compor serviços Web, ou seja, linguagens internas.

- **OWL-S**[7]: É uma ontologia de serviços que permite a descoberta, invocação, composição, interoperação e monitoramento da execução de maneira automática. OWL-S modela os serviços utilizando uma ontologia, que consiste em três partes: (1) *service profile*: descreve aquilo que o serviço requer dos usuários, (2) *service model*: especifica como funciona o serviço, e (3) *service grounding*: oferece informações sobre como usar o serviço.

- **Componentes Web**: Esta proposta trata os serviços como componentes para dar suporte a princípios básicos de desenvolvimento de software, tais como reutilização, especialização e extensão. A ideia principal é encapsular a lógica de informação composta dentro de uma definição de classe, o que representa um componente Web. A interface pública de um componente Web pode então ser publicada e utilizada para a descoberta e reutilização.

- **Álgebra de Processos** : Tem como meta introduzir descrições muito mais simples do que outras propostas, e modelar os serviços como processos móveis para garantir a verificação de propriedades, tais como segurança e gerenciamento de recursos. A teoria de processos móveis está baseada no *Pi-Calculus*, no qual a entidade básica é o processo, que pode ser um processo vazio, ou uma escolha entre várias operações de entrada/saída, uma composição paralela, uma definição recursiva ou uma invocação recursiva.

- **Redes de Petri**: As Redes de Petri são uma proposta de modelagem de processos bem estabelecida. Uma rede de Petri é um grafo orientado, conectado e bipartido, no qual os nós representam localizações e transições, e existem *tokens* que ocupam os estados. Os serviços podem ser modelados como redes de Petri, atribuindo transições para métodos e localizações para os estados. Cada serviço tem uma rede de Petri atribuída que descreve o comportamento e possui uma localização de entrada e uma de saída.

- **Inspeção de Modelos**: Uma inspeção de modelos é usada para verificar formalmente sistemas concorrentes de estado finito. A especificação do sistema é descrita usando lógica temporal, e para saber se a especificação é correta, realiza-se um teste no modelo. Pode-se aplicar a inspeção de modelos para a composição de serviços Web, verificando a corretude da especificação dos serviços. Dentre as propriedades que podem ser verificadas estão a consistência dos dados e a satisfação de restrições de negócio.

- ***Roman Model* (Ações com modelo de processos baseado em autômatos)**: Com esta proposta, os serviços são modelados de uma forma mais abstrata, baseada na noção de atividades. Basicamente, há um alfabeto (finito) de nomes de atividades, mas não é modelada a estrutura interna (entradas, saídas ou interação com o mundo).

- **Máquinas de Mealy (máquinas de estado finito)**: São máquinas de estado finito com entradas e saídas. Os serviços se comunicam através do envio de mensagens assíncronas, onde cada serviço tem uma fila. Um "observador" global faz acompanhamento de todas as mensagens. A conversa é vista como uma sequência de mensagens.

2.4 Internet do Futuro: A Internet das Coisas

Atualmente, a maioria de conexões à Internet no mundo são realizadas por dispositivos utilizados directamente por pessoas, tais como o computadores e telefones celulares. A principal forma de

[7]OWLS foi anteriormente conhecido como DAML-S

comunicação é de humano a humano. Em um futuro não muito distante, todos os objetos estarão conectados. As "coisas" poderão trocar informações por si mesmas e o número de "coisas" conectadas à Internet será muito maior do que o número de "pessoas", e os humanos podem tornar-se a minoria como geradores e receptores de tráfego. O mundo físico e o mundo da informação se juntarão e misturarão [TW10].

O futuro não será de pessoas falando com pessoas,e nem de pessoas acessando à informação. O futuro será usar máquinas para se comunicarem com outras máquinas em nome de pessoas. Nós estamos entrando em uma nova era de computação ubiqua, estamos entrando na era da "Internet das Coisas" (IoT) em que serão realizadas novas formas de comunicação entre pessoas e coisas, e entre coisas entre si. Assim, uma nova dimensão é adicionada ao mundo das tecnologias de informação e comunicação: a qualquer momento, qualquer lugar de conexão para qualquer um, e conectividade para qualquer coisa [TW10]. A Figura 2.6 mostra essa nova dimensão.

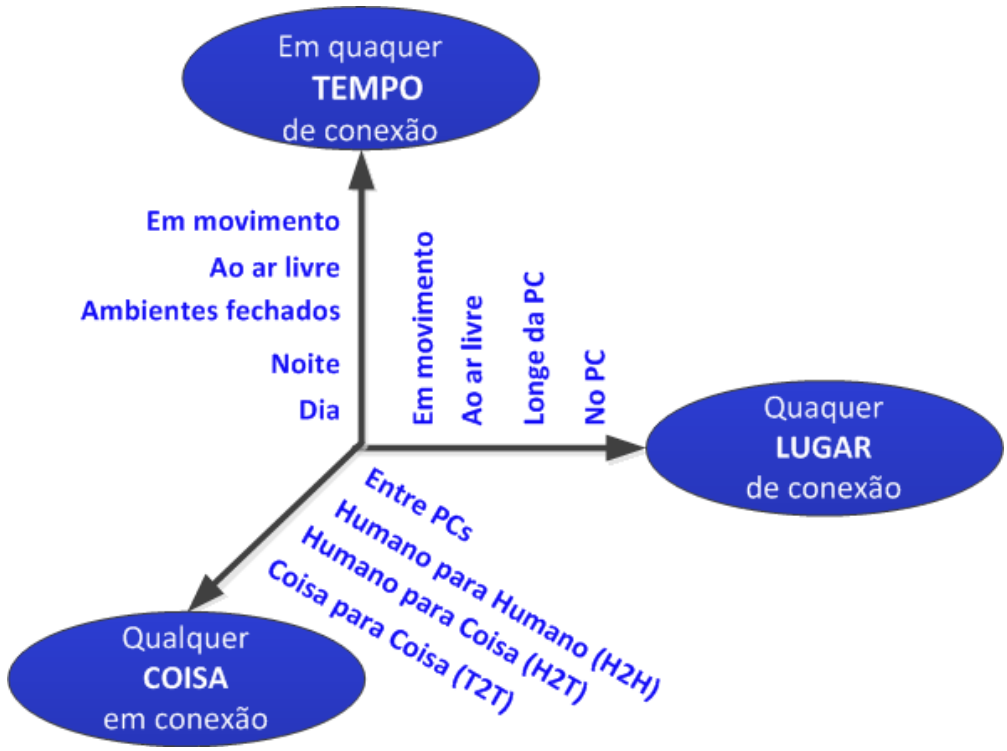

Figura 2.6: *Uma nova dimensão nas tecnologias de Informação [PTDL07]*

Não há nenhum padrão para definir a "Internet das Coisas". Considerando a funcionalidade e a identidade como central, a IoT pode ser definida como "Coisas que possuim identidades e personalidades virtuais desempenhando-se em espaços inteligentes utilizando interfaces inteligentes para se conectarem e comunicarem dentro de contextos de tipo social, de entorno e de usuário". Uma diferente definição com foco na integração poderia ser formulada como "Objetos interconectados com um papel ativo no que pode ser chamado de "Internet do Futuro" " [TW10]. Do ponto de vista conceitual, a "Internet das coisas" baseia-se em três pilares que estão relacionados com a capacidade dos objetos inteligentes para: (i) ser identificáveis (toda coisa se identifica por si mesma), (ii) para se comunicar (toda coisa se comunica) e (iii) para interagir (toda coisa interage), construindo redes de objetos interconectados, ou com usuários finais ou outras entidades na rede [MSDC12].

De uma perspectiva de nível de serviço, a questão principal tem a ver o como integrar (ou compor) as funcionalidades e/ou recursos fornecidos por objetos inteligentes (em muitos casos em formas de fluxos de dados gerados) dentro de serviços. Isto requer a definição de: (i) arquitecturas e métodos para "para virtualizar" objetos criando um padrão para representar objetos inteligentes no domínio digital, capaz de ocultar a heterogeneidade dos dispositivos/recursos e (ii) métodos para a integração e composição contínua de recursos/serviços de objetos inteligentes em serviços de valor agregado para usuários finais [MSDC12].

2.4.1 Tecnologias e Tendências

A "Internet das Coisas" é amplamente utilizado para se referir a: (1) o resultado de interconectar objetos inteligentes em uma rede global por meio de tecnologias de Internet estendidas, (2) o conjunto de tecnologias de suporte necessárias para concretizar essa visão (incluindo, por exemplo, RFID [8], sensores/atuadores, dispositivos de comunicação máquina a máquina, etc) e (3) de conjuntos de aplicações e serviços, aproveitando essas tecnologias para abrir novos negócios e oportunidades de mercado [MSDC12].

A partir de uma perspectiva de comprimento, a tendência de desenvolvimento da "Internet das Coisas" inclui três etapas: inteligência embarcada, conectividade e interação. Atualmente, existe inteligência embarcada em diversos dispositivos e domínios (sistemas de vôo, domésticos para casa, sistemas de misseis, entre outros) que podem atuar automáticamente. Porém, apesar de esses dispositivos sejam inteligentes, eles só trabalham sozinhos e localmente, e não há nada relacionado com a "rede".

Dessa maneira, o próximo passo seria conectar ou comunicar esses dispositivos inteligentes. No entanto, do ponto de vista de dispositivos inteligentes conectados, esses dispositivos não são inteligentes porque são apenas dotados com recursos de agentes e todas as ações são pré-concebidas por humanos, eles são inteligentes porque eles estão conectados. As "coisas" podem ser conectadas com fio ou sem fio. Na Internet das Coisas, a conexão sem fio será a principal maneira. Com base na infra-estrutura existente, há muitas maneiras de conectar uma "coisa": RFID, ZigBee, WPAN, WSN, DSL, UMTS, GPRS, WiFi, WiMax, LAN, WAN, 3G, etc. Conectar coisas inteligentes torna possível a interação.

Mesmo que possa se conectar qualquer "coisa", não significa que as coisas possam se comunicar por si mesmas. Assim, novas "coisas" inteligentes devem ser criadas, as quais podem processar informação, se auto-configurar, se auto-manter, se auto-reparar, fazer decisões independentes e, eventualmente, desempenhar um papel ativo na sua própria remoção. As coisas podem interagir, trocar informações por elas mesmas. Assim, a forma de comunicação irá mudar de humano-humano para a coisa-humana e coisa-coisa. Novos aplicativos de negócios deveriam ser criados e podem melhorar a inovação e desenvolvimento da Internet das Coisas. A Figura 2.7 mostra uma tendência de desenvolvimento aproximada da Internet de Coisas.

[8] RFID: Identificação por radiofrequência do inglês "Radio-Frequency IDentification"

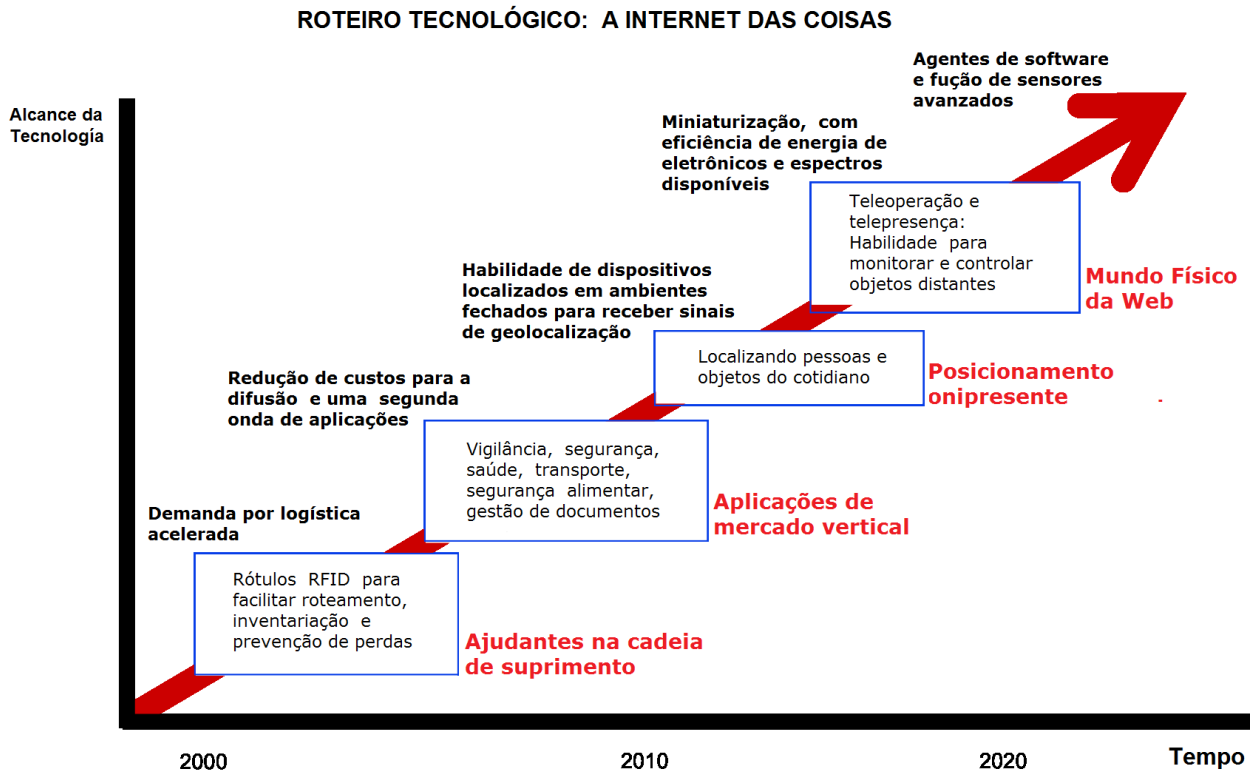

Figura 2.7: *Tendências no Internet do Futuro [TW10]*

Capítulo 3

Coreografias de Serviços Web

3.1 Definição

Como resumido no Capítulo 2, orquestração e coreografia são dois tipos de composição de serviços. Não existe um consenso acerca da definição formal de uma coreografia de serviços Web [Rt05]. O conceito, junto com os padrões, ferramentas e a tecnologia que a suportam, têm evoluído junto com a maturidade da SOA e dos serviços Web [CDMV09].

Em [Lie11], são apresentadas várias definições de "coreografia de serviços" agrupadas segundo camadas de domínios. Na camada de "Composição de serviços e coordenação" uma coreografia de serviços é definida como uma forma de composição de serviços na qual o protocolo de interação entre os diversos serviços participantes é definida de uma perspectiva global. Em [BDO05], apresenta-se uma definição genérica (independente do domínio), na qual uma coreografia de serviços é um processo colaborativo que envolve múltiplos serviços e em que a interação entre eles é vista de uma perspectiva global.

A definição informal de coreografia de serviços pode ser resumida como [CHY07]:

> "Os dançarinos dançam seguindo um cenário global, sem um único ponto de controle".

Isto é, em tempo de execução, cada participante em uma coreografia executa sua parte (chamada de papel), de acordo com o comportamento dos outros participantes. O papel da coreografia especifica o comportamento esperado na troca de mensagens dos participantes, em termos do sequenciamento e tempo das mensagens que eles consomem e produzem [SBFZ07].

Em [ADHR05] é apresentada uma definição de uma "coreografia real" sem fazer uma distinção entre uma coreografia e uma orquestração, porque ambos os termos são utilizados para se referir ao problema de acordo de um processo comum. Em [ADHR05], assume-se que uma coreografia define colaborações entre participantes que interagem, isto é, que no processo de coordenação dos serviços Web interconectados, todos os participantes precisam estar de acordo. A Figura 3.1 mostra essa noção de coreografia, onde os participantes interagem entre si sem um ponto de controle único. O acordo do ponto de vista global da coreografia está baseado no comportamento observável dos participantes, e o comportamento interno de cada um deles está especificado por meio de uma orquestração (BPEL).

3.1.1 Padrões e Linguagens

Diversas propostas para definir e especificar coreografias têm surgido. A seguir mostra-se as definições de coreografia de serviços segundo alguns dos padrões mais utilizados.

- **WSCI**[1]: A sua especificação não tem uma definição explicita de coreografia de serviços, mas a palavra coreografia(e as palavras derivadas) aparece no documento, no glossário define o que uma "coreografia" (mas o descreve como elemento da linguagem WSCI) descreve dependências

[1]WSCI (Web Services Choreography Interaction): www.w3.org/TR/wsci/

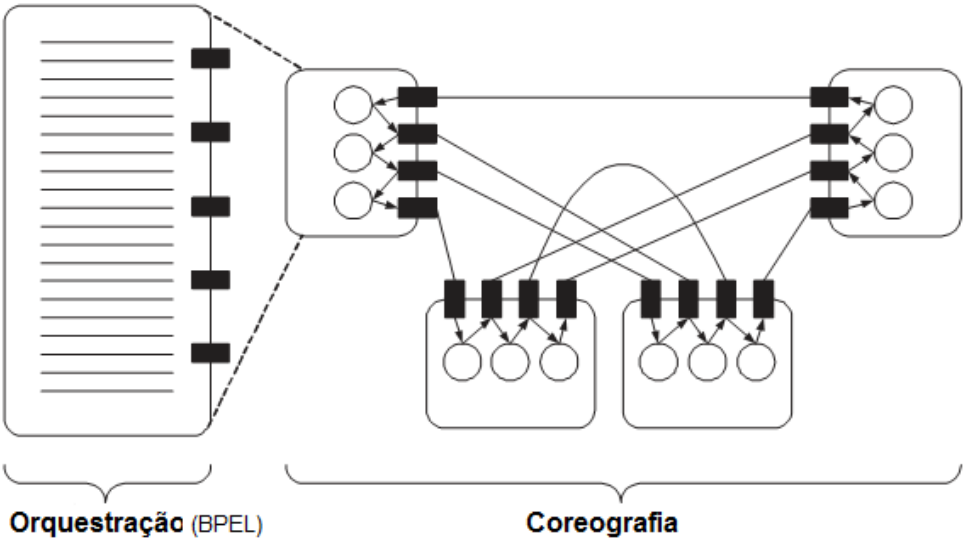

Figura 3.1: *A coreografia define colaborações entre participantes que interagem entre si [ADHR05]*

temporais e/ou lógicas entre atividades; define também o que é "coreografia de mensagens" (message choreography) como a ordem na qual as mensagens podem ser enviados ou recebidos em uma dada troca de mensagens, as regras que governam tal ordem e os limites da troca de mensagens(quando começam e quando terminam).

- **WS-CDL**: Em [W3C05] define a coreografia como um contrato multi-participante desde uma perspectiva Global. Na sua especificação, também não faz uma definição explicita de coreografia de serviços web, mas descreve por exemplo a camada de coreografia(choreography layer) como o componente que descreve colaborações entre participantes, definindo desde um ponto de vista global seus comportamentos observáveis comuns e complementários, onde a troca de informação acontece, quando as regras ordenadas e acordada em conjunto são satisfeitas.

- **BPMN**: No documento da especificação de BPMN 2.0 [2] define uma coreografia como uma seqüência ordenada de troca de mensagens B2B (Negócio a Negócio) entre dois ou mais participantes. Uma coreografia não possui um controlador central, nem uma entidade responsável e nem um observador do processo.

- **BPEL4Chor**[DKLW07]: Uma coreografia descreve a troca de mensagens entre serviços desde a perspectiva de um observador quem é capaz de ver as interações todas e o fluxo de dependências .

- **Let's Dance**[ZBDH06]: Em uma perspectiva global (coreografia), as interações são descritas desde a perspectiva de uma coleção de serviços(abstraídos como papéis), isto é útil quando acontece a comunicação acerca de como os serviços devem se comportar para corretamente interagir uns com os outros.

Finalmente, existem estudos nos quais se define formalmente uma coreografia de serviços Web, por exemplo em [ZWPZ10], [SBFZ07], [LW10] entre outros.

3.1.2 Orquestração e Coreografia de serviços

Ambas as abordagens fundamentam a área geral da engenharia de software orientada a serviços, e assim, representam diferentes escolhas de desenho na hora implementar sistemas orientados a

[2] BPMN 2.0: http://www.bpmn.org/

serviços [Ros09]. Depende do cenário concreto, para determinar, se usar uma abordagem explícita de "cima para baixo" *top-down* para uma coreografias e derivar orquestrações participantes, ou se utilizar uma abordagem *bottom-up* é suficiente.

Os fluxos de trabalho (*workflows*) podem ser descritos usando WS-BPEL (ou outra linguagem de orquestração) ou podem ser descritos por uma linguagem de coreografia de serviços como o WS-CDL, mas a escolha depende do que se quer conseguir. É frequente o caso em que as duas abordagens são necessárias. Por exemplo, um fluxo de trabalho pode ser descrito como uma coreografia, e este pode se realizar como um conjunto de fluxos de trabalho executáveis, que são enlaçados por desenho. Neste cenário os serviços podem ser orquestrados (usando por exemplo WS-BPEL), de modo que o comportamento externo observável é definido (e gerado) a partir de uma coreografia acordada (usando por exemplo WS-CDL). Assim, a coreografia é uma guia para o sistema todo, e a orquestração é uma maneira para realizar o sistema sem a introdução de um intermediário de serviços geral [Rt05].

A composição de serviços possui três pontos de vista [BDO06]:

- **Interface de Comportamento**: Chamada também de processos abstratos em BPEL e "Perfil do Protocolo de Colaboração" em ebXML [OAS05]. Este ponto de vista captura as dependências de comportamento entre as interações, nas quais um serviço individual pode cumprir o que se espera que ele cumpra. Em [BDO06], distinguem-se dois tipos de "Interfaces de Comportamento": "Interfaces Fornecidas"(isto é, "como está") que informam o que os serviços fornecem atualmente e as "Interfaces Esperadas" (isto é, "para ser") que informam o que espera-se que os serviços forneçam em uma dada configuração.

- **Orquestração**: Chamada também de processos executáveis em BPEL. Trata com a descrição de interações as quais um serviço dado pode se comunicar com outros serviços, assim como os passos internos entre essas interações.

- **Coreografia**: Chamada também de modelo global em WSCI e WS-CDL[3], e colaboração multi-participante em ebXML [OAS05]. Captura processos colaborativos envolvendo múltiplos serviços, especialmente suas interações vistas de um ponto de vista global.

A Figura 3.2 mostra a relação em alto nível que existe entre orquestração e coreografia. Orquestração se refere a um processo de negócio executável que interage com serviços Web internos e externos. As interações acontecem no nível de mensagens. Elas incluem lógica de negócios e a ordem da execução de tarefas, e podem estender as aplicações e as organizações para definir um modelo de processos de longa duração, transacional e de várias etapas. A orquestração sempre representa o controle da perspectiva de um participante. A coreografia é mais colaborativa e permite que vários participantes estejam envolvidos para que descrevam sua parte na interação. A coreografia acompanha a sequência de mensagens de múltiplos participantes e fontes, ao invés de apenas um processo de negócio específico.

Como o objetivo desta pesquisa é propor mecanismos relacionados com coreografia de serviços Web, a partir deste ponto este texto focará neste tipo de composição.

3.1.3 Processos de Negócio

Um processo de negócio interorganizacional (IOBP) é um grupo organizado de atividades unidas realizadas por duas ou mais organizações para atingir um objetivo de negócio. A conseqüência disso, a modelagem e projeto de processos de negócios usados dentro de uma organização tem que ser melhorada e estendida para lidar com particularidades de negócios interorganizacionais [BA11].

Do ponto de vista de processos de negócios, as coreografias descrevem o comportamento externamente observável de uma entidade de negócios em processos de negócio inter-organizacionais. A idéia por trás das coreografias está relacionado ao fato de que as entidades de negócios freqüentemente consideram seus processos intra-organizacionais como ativos (por exemplo, visão baseada

[3]WS-CDL: Web Service Choreography Description Language

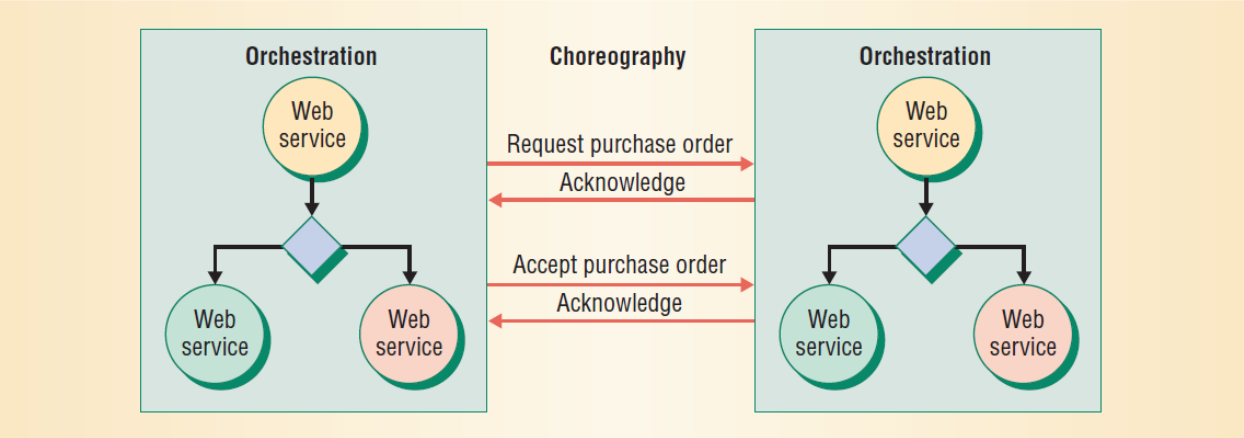

Figura 3.2: *Orquestração Vs. Coreografia [Pel03]*

em recursos em gestão estratégica) [MH05]. No entanto, as empresas querem se beneficiar de fortes colaborações com os seus parceiros (participantes). Assim, as linguagens de coreografias são maneiras de definir regras de colaborações entre vários participantes sem revelar as suas operações internas [MH05].

Além disso, a modelagem de processos de negócio inter-organizacionais envolve novos desafios, principalmente, a capacidade de lidar com autonomia, privacidade, heterogeneidade e com suporte de coordenação por meio de contratos. Nesse contexto, as coreografias são uma abordagem para modelar processos (ou *workflows*) inter-organizacionais do ponto de vista global (ver Figura 3.3). Nesse esquema os processos processo ficam descritos de maneira "privada" (orquestração, interna ou executável), "pública" (coreografia local, abstrata ou vista), e "colaborativa" (coreografia global, inter-organizacional ou *cross-organizacional*), a fim de separar melhor a densidade de informação de diferentes áreas.

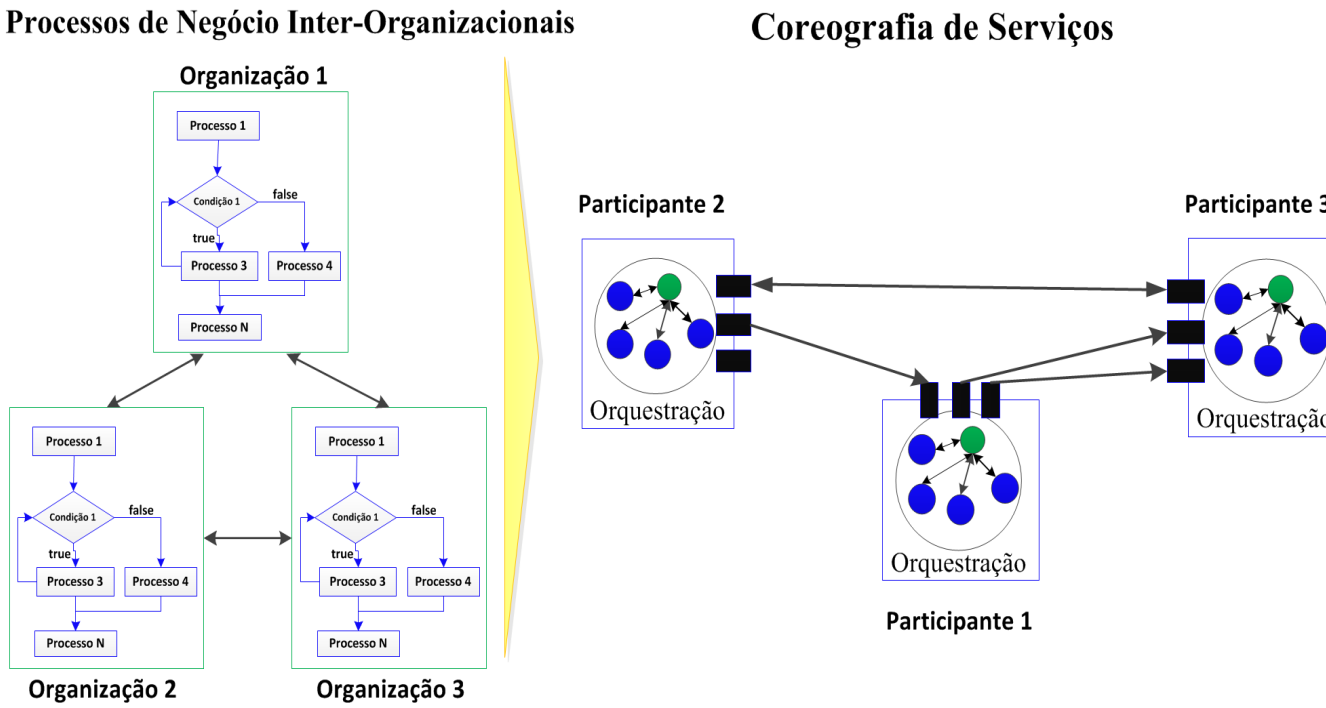

Figura 3.3: *Coreografia como solução para modelagem de processos de negócio inter-organizacionais*

3.2 Elementos de um Modelo de Coreografia

Segundo [SBFZ07], uma linguagem de modelagem de coreografias fornece os meios para definir um modelo de coreografia, isto é, coreografias, implementação de serviços e semântica, incluindo mecanismos para comparar a coreografia com o comportamento global das implementações dos seus serviços. Assim, pode-se entender informalmente uma linguagem de modelagem de coreografias L como uma coleção de modelos de coreografias:

$$L = \{(C, I) \mid C \textit{ é uma coreografia e } I \textit{ a implementação mediante serviços}\}$$

Onde C é a coleção de coreografias e I é a coleção de implementações de serviços. Em tempo de execução, cada participante é responsável pela correta execução do seu papel na coreografia, isto é, seu comportamento esperado na troca de mensagens em relação ao comportamento dos outros participantes. Quando os participantes realizam ou executam seus papéis segundo o acordo comum entre eles, isto é chamado de *enactment* da coreografia [MPRW10], [Bul08].

Uma coreografia pode ser definida utilizando dois tipos básicos de elementos [SBFZ07]:

1. Um conjunto de ações observáveis que acontecem nos serviços individuais (localmente).
2. Conjunto de restrições (globais) de sequência de atividades em (1).

As ações observáveis tipicamente são de dois tipos: Ações por meio de mensagens para se comunicar com outros serviços, e as ações ("atividades") locais, que são realizadas nos serviços individuais independentemente de outros serviços. As restrições de sequência restringem a ordem das ações dos participantes ou serviços. Além disso, os serviços devem utilizar um modelo de mensagens para se comunicar, que tipicamente são os modelos síncrono e assíncrono. No caso do modelo assíncrono, é comum utilizar uma fila *FIFO*[4] (de tamanho finito ou ilimitado) no receptor para armazenar por ordem de chegada as mensagens que ainda não foram consumidas [MPRW10]. Dadas estas definições, os modelos de coreografia de serviços têm que lidar com os seguintes problemas [Bul08], [MPRW10]:

- **Ausência de *deadlock***: Um *deadlock* ocorre quando o *enactment* de uma coreografia atinge um estado que, (1) não é o final e que (2) não pode deixar sem violar a ordem das mensagens na coreografia. As análises de coreografias para a ausência de *deadlocks* geralmente são construídas sobre técnicas de verificação de modelos [MPRW10].
- **Conformidade**: Um participante está em conformidade com uma coreografia se seu protocolo de negócio, isto é, o atual comportamento da troca de mensagens do participante como é percebido pelos outros, é equivalente ao papel especificado por esse participante na coreografia [MPRW10].
- ***Realizability***: *Realizability* indica se *peers* podem ser gerados a partir de uma especificação de coreografia, de maneira que as interações dos *peers* gerados encaixem exatamente com a especificação da coreografia [BR09]. Em outras palavras, *realizability* é a possibilidade de que uma especificação de coreografia possa ser implementada (realizada) pelos serviços ou participantes [SBFZ07].
- **Síntese**: A síntese de uma coreografia denota a derivação de processos de orquestração executáveis a partir de uma especificação de coreografia [ZWPZ10]. Uma síntese ideal pode tomar um documento WS-CDL válido para derivar um conjunto de processos de orquestração, os quais são semanticamente equivalentes à coreografia original dada. Portanto, o problema da síntese é decidir se uma coreografia pode ser realizada por algumas orquestrações (referidas como implementáveis) e sintetizar uma combinação de orquestrações se for possível [SLD+10].

[4] FIFO: Acrônimo em inglês para Primeiro em Entrar Primeiro em Sair.

- **Sincronização**: A análise da sincronização visa avaliar os efeitos da comunicação síncrona e assíncrona para melhorar a eficiência da interação. Assim, um conjunto de *peers* que se comunicam assincronamente são sincronizáveis, se seu conjunto de mensagens não muda quando a comunicação assíncrona é substituída por uma comunicação síncrona [MPRW10].

- **Compatibilidade**: A compatibilidade de uma coreografia de serviços é a capacidade de um conjunto de serviços Web interagir por meio de troca de mensagens de uma forma confiável. O conjunto de serviços compatíveis não somente dependem da sequência de mensagens, mas também dependem de propriedades quantitativas como o tempo [GG11].

- **Reconfiguração Dinâmica**: Tais reconfigurações correspondem à adição ou remoção de algumas interações em tempo de execução por conta de eventos como perda do serviço, extensão de funcionalidades, substituição de um serviço, entre outros [SR09]. Assim, é importante formalizar a reconfiguração para verificar se um conjunto de *peers*, podem ser reconfigurados com relação a uma segunda especificação de coreografia, a qual consiste de uma extensão (adição de algumas interações) ou uma simplificação (remoção de algumas interações) da coreografia original. Se essas reconfigurações são possíveis, novos *peers* são gerados.

As linguagens de coreografia, segundo [SBFZ07], podem ser divididas em três categorias baseadas em seus arcabouços associados: autômatos de estado finito, redes de *Petri* e álgebra de processos.

3.2.1 Modelos baseados em Autômatos

Os modelos baseado em autômatos representam a coreografia e a implementação de serviços como uma maquina de estado finito, isto é, especificam uma coreografia por meio de estados e transições. Desta maneira, capturam explicitamente uma instantânea da execução do serviço composto como um estado, e os comportamentos(locais e globais) podem ser capturados como uma sequência de estados, nos quais cada estado de transição está associada a uma mensagem ou uma atividade [SBFZ07].

Este grupo de linguagens de modelagem de coreografias incluem: protocolos de conversação, serviços *Mealy* [BFHS03] , diagramas de colaboração UML [BF08], e o modelo de composição de serviços Colombo [BCG+05]. A linguagem *Let's Dance* [ZBDH06] também pertence a este grupo, e fornece de um conjunto de primitivas de restrição de sequências para permitir especificar uma coreografia com uma notação gráfica.

3.2.2 Modelos baseados em Redes de Petri

São amplamente utilizados para modelar, entre outras coisas, fluxos de controle, e por conta disso, são candidatos adequados para linguagens de modelado de coreografias. Por exemplo, IPN (*Interaction Petri Net*), uma linguagem de modelado de coreografias baseada em Petri, trata uma troca de mensagem como um disparo de uma transição. O uso de redes de Petri permite que conversações concorrentes sejam explicitamente separadas, ao contrário do que permitiria um diagrama de colaboração UML [SBFZ07]. Os serviços ou participantes são representados por "interfaces de comportamento", que são redes de Petri com lugares de entrada e saída. Os serviços ou participantes se comunicam com outros no modelo de mensagens síncronos, em lugar de uma fila FIFO, que utilizam as linguagens baseadas em autômatos.

3.2.3 Modelos baseados em Álgebra de Processos

São menos intuitivos e não possuem representação gráfica , e assim como os modelos baseados em redes de Petri, fornecem de um formalismo para a verificação das propriedades de uma especificação. Recentemente, existem várias propostas no desenvolvimento de linguagens de modelado de coreografias baseadas em álgebra de processos [SBFZ07], muitas dessas propostas estão baseada em cálculo Pi. O cálculo Pi é uma álgebra de processos que tem como intuito ser uma teoria formal para modelado de processos. Utilizam o modelo de mensagens síncronos na comunicação dos

participantes. Exemplos de linguagens deste tipo são XLANG, WS-CDL (e as baseadas nela, tais como CDL [PSW+07] e *Chor* [ZXCH07]), o modelo "Bologna" [SBFZ07], entre outras.

3.3 Linguagens de Coreografias de Serviços

As linguagens de coreografia de serviços podem ser categorizadas utilizando dois critérios [DKB08]: as independentes de implementação e as específicas de implementação. As linguagens independentes de implementação são utilizadas principalmente para descrever processos da perspectiva de negócios. A definição de formatos de mensagens concretas ou protocolos de comunicação é realizada no escopo das linguagens específicas de implementação.

Basicamente existem duas abordagens de modelagem para linguagens de coreografia de serviços [BWR09]: modelos de interação e os modelos de interconexão. Os modelos de interação usam interações atômicas como blocos de construção básicos (a troca de mensagens entre os participantes) e, os fluxos de dados e de controle estão baseados nas dependências entre essas interações de uma perspectiva global. Os modelos de interconexão definem o fluxo de controle por participante ou por papel do participante, e as respectivas atividades de envio e recepção são conectadas usando fluxos de mensagens, representando deste modo as interações. A Figura 3.4 mostra a categorização das linguagens de coreografia segundo os critérios e as abordagens citados. A seguir, essas linguagens são descritas resumidamente.

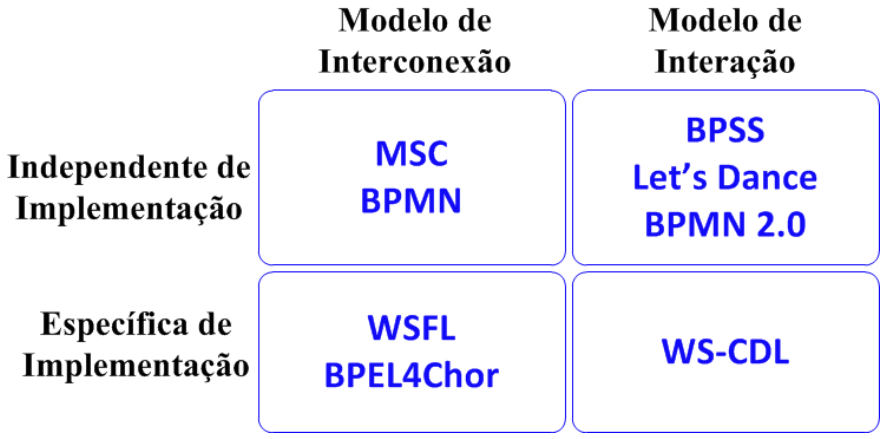

Figura 3.4: *Categorização de linguagens de coreografias de serviços Web (baseada em [Eng09])*

O BPMN é uma linguagem de modelagem gráfica e é o padrão OMG [5] para modelar processos. Esta linguagem realiza a distinção explícita entre o fluxo de controle e o fluxo de mensagens. Todas as atividades conectadas por meio de fluxos de controle pertencem ao mesmo processo e o fluxo de mensagens é utilizado para interconectar processos diferentes. O MSC (Tabela de Sequência de Mensagens) é uma linguagem mais adequada para modelar sequências de interações simples ao invés de coreografias complexas, já que não suporta ramos condicionais, ramos paralelos e nem iterações [DKB08].

O protocolo BPSS [6] pode definir coreografia e comunicação entre serviços baseadas em XML, e pertence ao grupo de especificações do *ebXML* da OASIS. Sua principal desvantagem é suportar somente coreografias entre dois participantes. *Let's Dance* é uma linguagem de modelagem gráfica e, assim como BPMN, está focado na análise de negócios e captura de requisitos, em lugar de detalhes de implementação. *Let's Dance* suporta mais cenários de coreografias do que BPMN [ZBDH06] e suporta também todos os padrões de coreografia comuns [BDH05]. BPMN 2.0 [7] é a versão atual do BPMN e ela introduz dois novos tipos de diagrama: diagramas de coreografia e diagramas de

[5] OMG: Object Management Group
[6] BPSS: *Business Process Specification Schema*
[7] BPMN 2.0: http://www.omg.org/spec/BPMN/2.0/

colaboração. O primeiro permite modelar fluxos entre distintas interações, enquanto que o segundo fornece uma visão geral em cenários complexos com diversos participantes.

O WSFL é uma linguagem baseada em XML que consiste de várias vistas locais (também chamados de modelos de fluxo) para especificar processos de negócio executáveis, e um modelo global para especificar colaborações e interações entre os participantes de negócio. O BPEL4Chor [DKLW07] é uma proposta que adiciona uma camada sobre o BPEL para mudar de uma linguagem de orquestração para uma linguagem de coreografia completa. No BPEL4Chor os processos BPEL abstratos são utilizados para descrições de comportamento entre participantes, as quais são unidas por meio de mensagens formando uma topologia de participantes. Assim como Let's Dance, o BPEL4Chor suporta todos os padrões de coreografia comuns [DKB08].

O WS-CDL é uma recomendação candidata pela W3C para se tornar a linguagem padrão para coreografias de serviços Web. Ela é baseada em *Pi-calculus*, uma linguagem formal que permite a descrição de algoritmos de programação concorrente. As dependências entre as interações são definidas por meio de um conjunto de construções de fluxo de controle que são difíceis de mapear para processos BPEL, sendo este ponto um dos aspectos mais criticados desta linguagem [BDO05]. A WS-CDL é muito utilizada em vários trabalhos sobre coreografias de serviços Web, e inclusive foram propostas linguagens derivadas dela, tais como *Chor* [ZXCH07] e WSCDL+ [KWH07].

3.4 Ferramentas e Arcabouços

Existem várias ferramentas e arcabouços que têm sido desenvolvidos para apoiar em diversos aspectos da coreografia de serviços, tais como modelagem, verificação e validação, simulação, síntese e realização, mapeamento para processos executáveis (como BPEL) e *enactment*. Alguns arcabouços abrangem muitos desses aspectos dentro de uma metodologia de desenvolvimento, por exemplo, o projeto Savara [SAV09] do JBoss. A seguir, são resumidas algumas dessas ferramentas e arcabouços.

LTSA-WS [FUMK06], *WS-CDL Eclipse* [Dou05] e Pi4SOA [ZTW+06] foram algumas das primeiras implementações para suportar modelagem, simulação e verificação de coreografias de serviços Web baseadas em WS-CDL. Estas três ferramentas são indicadas na especificação WS-CDL [Web05].

WS-CDL Eclipse é uma ferramenta baseada em Eclipse para produzir, visualizar e simular coreografias de serviços Web a partir de um documento WS-CDL. Pi4SOA fornece uma ferramenta baseada em Eclipse para modelar coreografias, além de possibilitar a geração de serviços a partir de um documento WS-CDL. Atualmente, o desenvolvimento de Pi4SOA está descontinuado, e nesse contexto apareceu o projeto Savara [SAV09]. O Savara faz parte da comunidade do JBoss e propõe uma nova metodologia de desenvolvimento chamada de arquitetura testável [Kum09].

Em [LK08] é apresentado um conjunto de ferramentas para suportar modelagem, verificação, validação e transformação para processos executáveis BPEL dentro do contexto do BPEL4Chor.

OpenKnowledge [BPB+09] é uma plataforma baseada em uma arquitetura ponto a ponto (P2P) para especificar e executar[8] coreografias de serviços. Cada *peer* representa um participante da coreografia. Esse *peer* é uma *wrapper* que separa o comportamento observável de um participante da sua implementação, a qual pode ser um serviço Web, um serviço composto, entre outros. Esta plataforma utiliza o LCC (*Lightweight Coordination Calculus*) [Rob05] como a linguagem executável baseada no *Process-Calculus* para especificar e executar as coreografias. A sua arquitetura é capaz de tratar com a heterogeneidade semântica dos seus participantes e a descoberta deles.

3.5 Coreografias em BPMN

3.5.1 BPMN

O BPMN (Notação de Modelagem de Processos de Negócio) é um padrão que surgiu como uma notação para capturar processos de negócio, especialmente no nível da análise de domínios e design

[8] Neste caso, o termo executar é válido, já que o LCC é uma linguagem executável de coreografias .

de sistemas alto nível. A notação herda e combina elementos a partir de uma série de anotações previamente propostas para modelagem de processos de negócios, incluindo a XPDL [9] e diagramas de atividades do UML.

O BPMN fornece uma notação gráfica para modelagem de processos de negócios, com ênfase em controle de fluxo. Ele define um Diagrama de Processos de Negócio (BPD), uma espécie de fluxograma que incorpora construções adaptadas para a modelagem de processos de negócios, tais como *AND-split*, *AND-join*, *XOR-split*, *XOR-join* e escolha diferida (baseada em eventos). Um BPD é composta de elementos BPMN. A Figura 3.5 mostra um subconjunto do núcleo de elementos de BPMN. Há objetos e fluxos de seqüência. Um objeto pode ser um evento, uma atividade ou um *gateway*. Uma fluxo de seqüência liga dois objetos em um BPD e mostra a relação do fluxo de controle (ou seja, a ordem de execução).

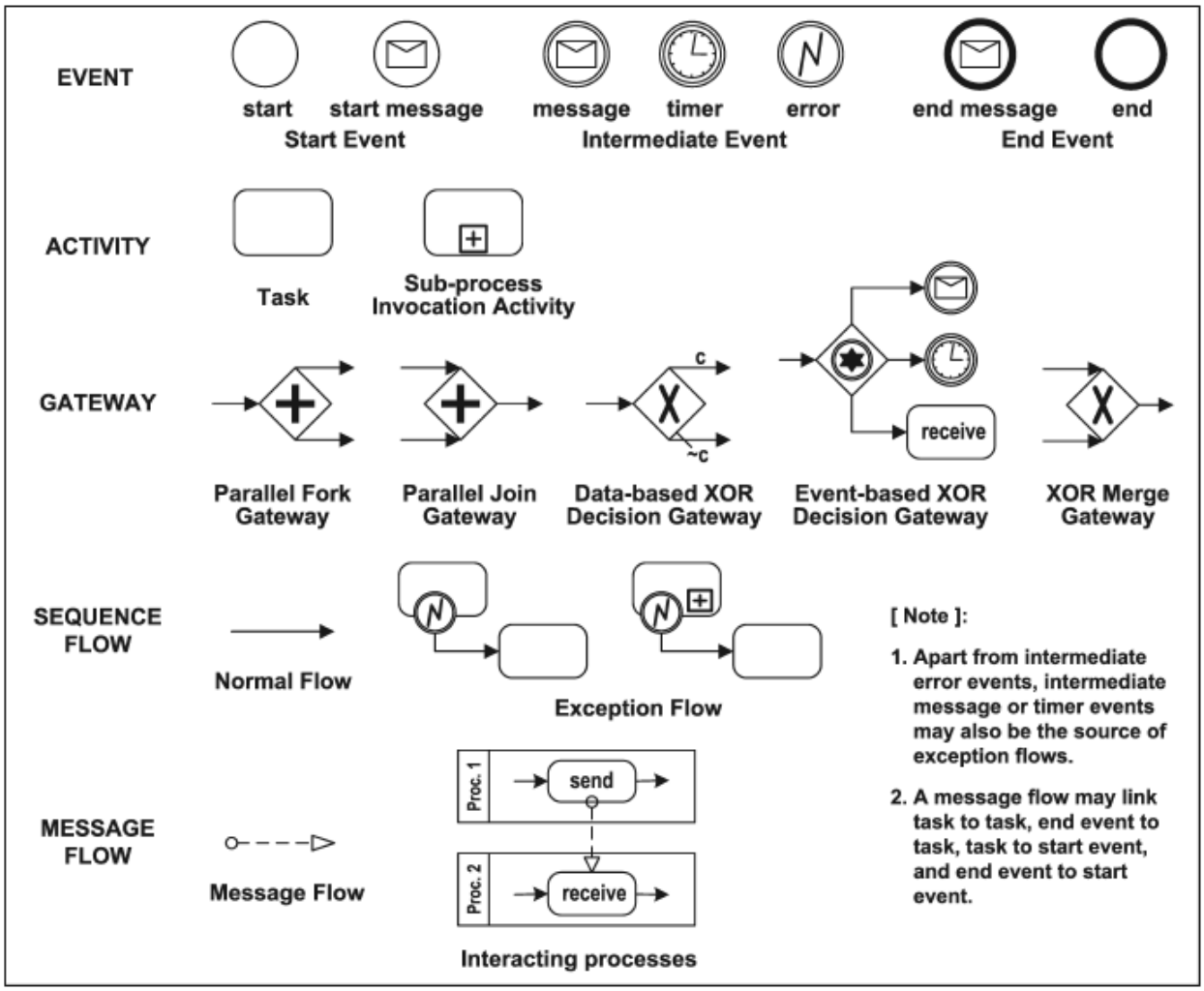

Figura 3.5: *Conjunto dos principais elementos do BPMN [DDO07]*

Um evento pode sinalizar o início de um processo (evento de inicio), o final de um processo de (evento de finalização), e pode também ocorrer durante o processo (evento intermediário). Um evento de mensagem é usado para enviar (não para o evento de início) ou receber (não para o evento de finalização) uma mensagem. Um evento temporizador (*timer*) indica um tempo ou data específico a ser atingido, e um evento de erro sinaliza um erro a ser detectado durante a execuçaõ de um processo.

Uma atividade pode ser uma tarefa ou um subprocesso. Uma tarefa é uma atividade atômica e significa trabalho a ser executado dentro de um processo. Há sete tipos de tarefa: serviço, de recebimento, de envio, de usuário, um *script* manual, e de referência. Por exemplo, uma tarefa de

[9] XPDL: Linguagem de Definição de Processo XML

recebimento é utilizada quando um processo aguarda por uma mensagem chegar de um participante externo. Um subprocesso é uma atividade composta que é definida como o fluxo de outras atividades. Existem subprocessos embarcados e subprocessos independentes. A diferença é que um subprocesso embarcado é parte de um processo, enquanto um subprocesso independente pode ser chamado por diferentes processos. Um subprocesso pode ser invocado via uma atividade de invocação de subprocessos.

Um *gateway* é uma construção de roteamento utilizada para controlar a convergência e divergência do fluxo de sequência. A seguir a descrição dos principais tipos de *gateways*:

- **De divisão em paralelo (*AND-split*)**: para a criação de fluxos de seqüência concorrentes.
- **De junção paralela (*AND-join*)**: para a sincronização simultânea de vários fluxos de seqüência.
- **De escolha XOR baseados em dados/eventos (*data/event-based XOR decision*)**: para a seleção de um conjunto de fluxos de sequência mutuamente excludentes. Onde a escolha está baseada ou nos dados do processo (*data-based*, isto é, *XOR-split*) ou de eventos externos (*event-based*, isto é, escolha diferida).
- **De junção XOR (*XOR-join*)**: para juntar um conjunto de fluxos de sequência mutuamente excludentes dentro de um único fluxo de sequência.

Em particular, um *gateway* de escolha XOR baseado em eventos deve ser seguida por alguma tarefa de recebimento ou evento intermediário para capturar condições de corrida baseado no tempo ou disparadores externos (por exemplo, o recebimento de uma mensagem de um participante externo). Para um maior entendimento e informação acerca dos elementos fornecidos por BPMN, consultar em [OMG11].

3.5.2 Coreografia de Processos

O BPMN é uma linguagem independente da implementação e, especialmente, permite a definição de coreografias interligando diferentes processos utilizando fluxos de mensagens. Essas coreografias são de alto valor, principalmente em contextos inter-organizacionais, onde diferentes parceiros ou participantes de negócios acordam seu comportamento de interação antes de interconectar seus sistemas de informação. Descrições de comportamento dos participantes são primeiro utilizadas como vistas da coreografia de acordo com a perspectiva de um participante individual. Depois disso, eles servem como especificações para a implementação de novos serviços ou a adaptação de serviços existentes [PDKL08].

Como já descrevido na Seção 3.3, existem duas maneiras de especificar coreografias de serviços, com modelos de interação e com modelos de interconexão. Em ambos os estilos de modelagem, os participantes interagem entre si e as actividades são conectadas. Por um lado, no modelo de interconexão conecta atividades de comunicação pertencentes a dois participantes. Assim, cada troca de mensagem (enviar/receber) é expressa usando uma conexão entre os participantes [KLW11]. Por outro lado, os modelos de interação expressam cada troca de mensagem (enviar/receber) como interações atômicas. Em geral, os termos "modelo de interconexão" e "modelo de interação" são derivadas da maneira como cada paradigma de modelagem expressa a troca de mensagens e não do fato de se as atividades são conectadas ou se são interações gerais entre os processos que são apresentados [KLW11].

Essas duas formas de especificar coreografias podem ser considerada como diferentes vistas de uma mesma coreografia [KLW11]. O padrão BPMN suporta a especificação de coreografias para ambos os modelos, sendo que só a partir do BPMN versão 2.0 o modelo de interação é suportado.

O Modelo de Interconexão

Os modelos de interconexão modelam cada participante como um processo separado, e pode prover um comportamento interno. Exemplos para linguagens de coreografia neste modelo são colaborações em BPMN 2.0, modelos com vários *pools* em BPMN 1.2, BPMN+, ou BPEL4Chor. O comportamento de cada participante pode ser expresso como um único elemento *pool* do BPMN ou um processo BPEL abstrato [KLW10]. Outras possibilidades adicionais para especificar o comportamento de um participante incluem a abordagem Open Net [KLW10].

Em BPMN 2.0 os modelos de interconexão são implementados por **diagramas de colaboração**. Estes diagramas consistem em dois ou mais *pools*, em que cada um deles representa um participante na colaboração global. Dentro de um *pool*, o comportamento do processo interno de um participante é modelado com elementos comuns do BPMN (por exemplo, atividades, eventos, e os fluxos de seqüência) [KLW11]. Alternativamente, os *pools* podem ser descritos como caixas pretas, isto é, o comportamento no interno de um processo é modelado.

A Figura 3.6 apresenta um exemplo de coreografia especificada de acordo com o modelo de interconexão. O exemplo é acerca de oferta de investimentos e está especificada utilizando a notação BPMN 1.2. Primeiro, um consultor financeiro (participante) oferece um produto a um cliente (outro participante). Posteriormente, o cliente tem 24 horas para decidir se aceita a proposta de investimento ou rejeita a proposta. Cada participante é representado por meio de **pools** nos quais se interconectam atividades.

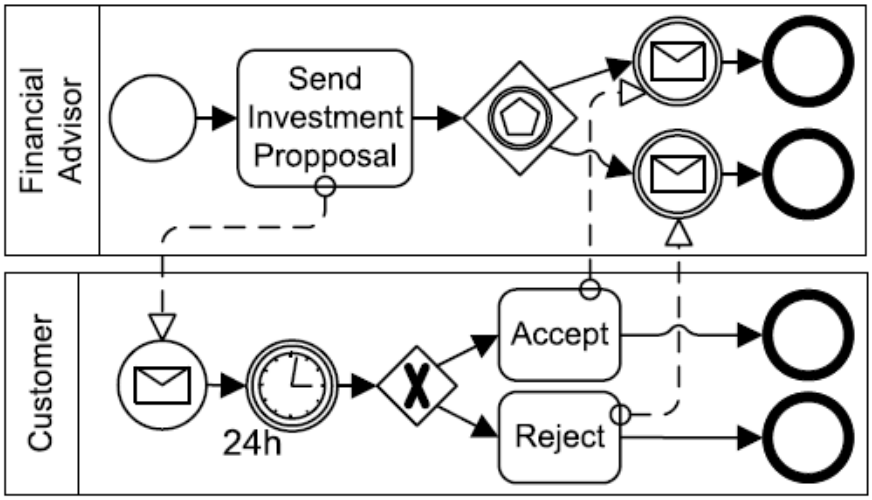

Figura 3.6: *Exemplo de modelo de interconexão de coreografias, oferta de investimento [KLW10].*

Os padrões de interação de serviços [BDH05] descrevem um conjunto de cenários recurrentes de coreografias. Eles variam de trocas de mensagens simples para cenários envolvendo múltiplos participantes e várias trocas de mensagens. Esses padrões podem ser usados para avaliar linguagens de coreografia. Embora o BPMN permita definir coreografias de acordo com o modelo de interconexão, ele apenas fornece suporte direto para um conjunto limitado de padrões de interação de serviços [DP07].

Além disso, esta abordagem de modelagem leva as dependências de fluxos de controle a serem redundantes e ter mais chances de processos incompatíveis. Um exemplo para essa incompatibilidade seria um provedor que aguarda a realização de um pagamento antes de entregar os bens adquiridos. O comprador, por outro lado, espera que as mercadorias possam ser entregues antes de realmente pagado eles. Ambos os parceiros iriam esperar indefinidamente - uma clásica situação de impasse (*deadlock*). Os modelos de interação evitam esses problemas por meio da descrição de dependências de fluxos de controle entre as interações [DB08]. Isso significa que uma dependência de fluxo de controle específico não é explicitamente atribuído a qualquer um dos participantes no modelo. Outra desvantagem de redundância é que os modeladores precisam de mais tempo para criar e entender os modelos. Em [DB08], verificou-se que o modelo de interação permite uma

rápida criação e compreensão para modeladores humanos.

3.5.3 O Modelo de Interação

O modelo de interação tem como bloco de construção de coreografias as interações atômicas entre participantes por meio de troca de mensagens. O modelo de interação para especificar coreografias é usado neste trabalho.

O bloco básico de construção nos modelos de interação são as "atividades" de interação. Essa atividades descrevem uma troca de mensagens entre dois participantes de maneira atômicas na especificação de uma coreografia. Os modelos de interação fornecem de uma visão de alto nível das colaborações e a maioria de linguagens de coreografia neste modelo ocultam o comportamento interno dos participantes [KLW10]. BPMN 2.0, iBPMN e BPELGold são linguagens que suportam o modelo de interação, das quais apenas o BPMN 2.0 é padrão. Para especificar ou modelar coreografias nesta abordagem utilizam-se "diagramas de coregrafias" suportados a partir de BPMN 2.0

Alguns elementos BPMN são comuns para os diagramas de processos normais e para os diagramas de coreografias (modelo de interação), bem como colaborações (modelos de interconexão). Tais elementos comuns incluem principalmente eventos, *gateways*, e fluxos de sequência e não incluem os "pools". As interações entre os participantes são representadas mediante atividades de coreografias (*choreography activity*). Uma atividade de coreografia representa um ponto no fluxo de uma coreografia, em que uma interação acontece entre dois ou mais participantes. Uma atividade de coreografia pode se referir a elementos como "subcoreografia", "chamada a outra coreografia" (*callchoreography*) e uma "tarefa de coreografia" (*choreography task*), das quais a última será levada em consideração nesta pesquisa.

Uma tarefa de coreografia é uma atividade atômica em uma coregrafia de processos. Representa uma interação em que uma ou duas trocas de mensagens acontecem entre dois participantes. A Figura 3.7 mostra uma tarefa de coreografia, em que as duas bandas representam os participantes, a banda em cor branca é do participante iniciador da interação e a banda escura é do participante dono da tarefa que está representada pela banda grande do meio.

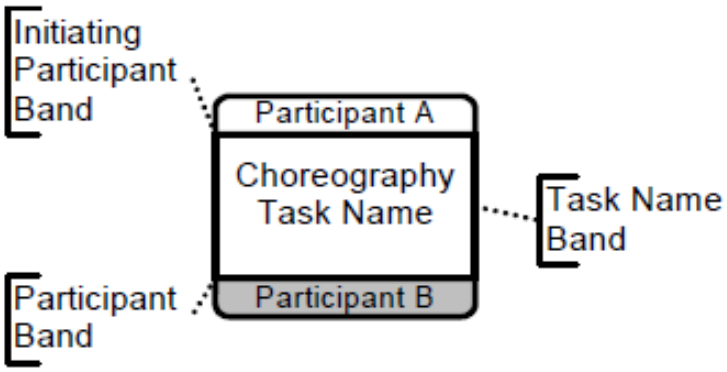

Figura 3.7: *Representação de uma tarefa de coreografia [NF10]*

As tarefas de coreografias são de um e de dois sentidos. Uma tarefa de coreografia de um sentido se mostra na Figura 3.7 e captura interações onde ocorrem apenas um envio de mensagem sem resposta, o que pode ser visualizado melhor no seu respectivo diagrama de colaboração (Figura 3.8). Uma tarefa de coregrafia de dois sentidos captura interações atômicas em que existe reposta. A sua representação contêm as duas mensagens enviadas, como mostrado na Figura 3.9 com seu respectivo diagrama de colaboração para visualizar melhor a interação. A mensagem de cor branca é a iniciadora (a partir do participante iniciador) e a mensagem de cor preta é a resposta.

A Figura 3.10 apresenta o modelo de interação equivalente do apresentado no modelo de interconexão (Figura 3.6). Nesse caso, o fluxo de controle é representado entre os *pools* e as atividades de mensagens locais foram substituídos por atividades de interações atômicas.

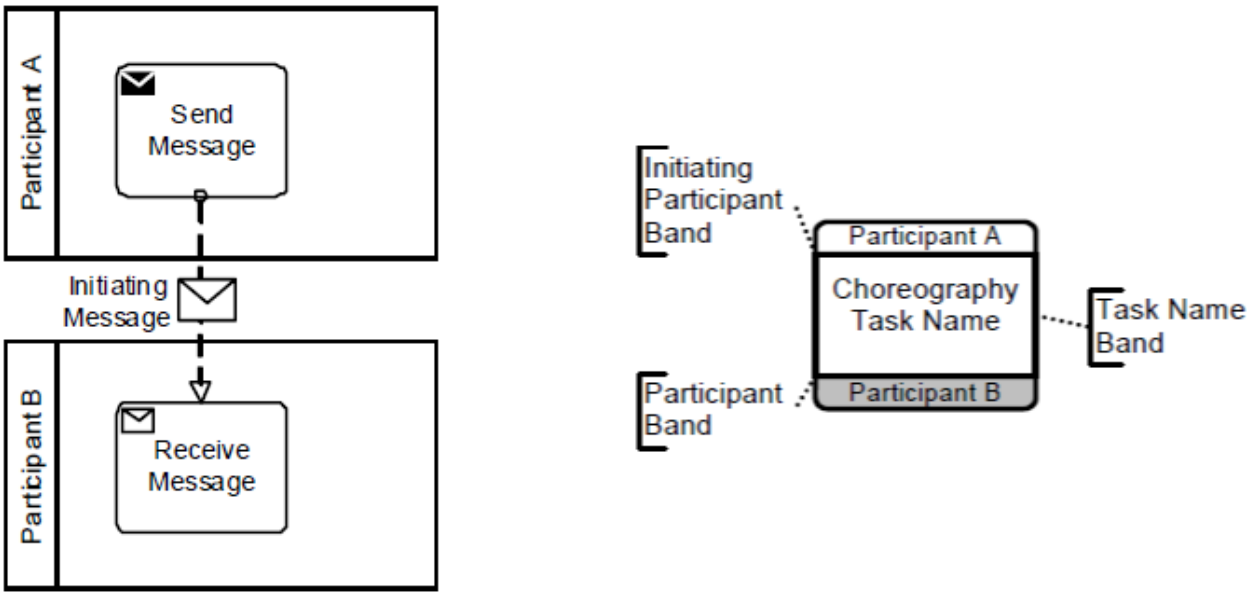

Figura 3.8: *Uma tarefa de coreografia simples e seu equivalente em diagrama de colaboração [NF10]*

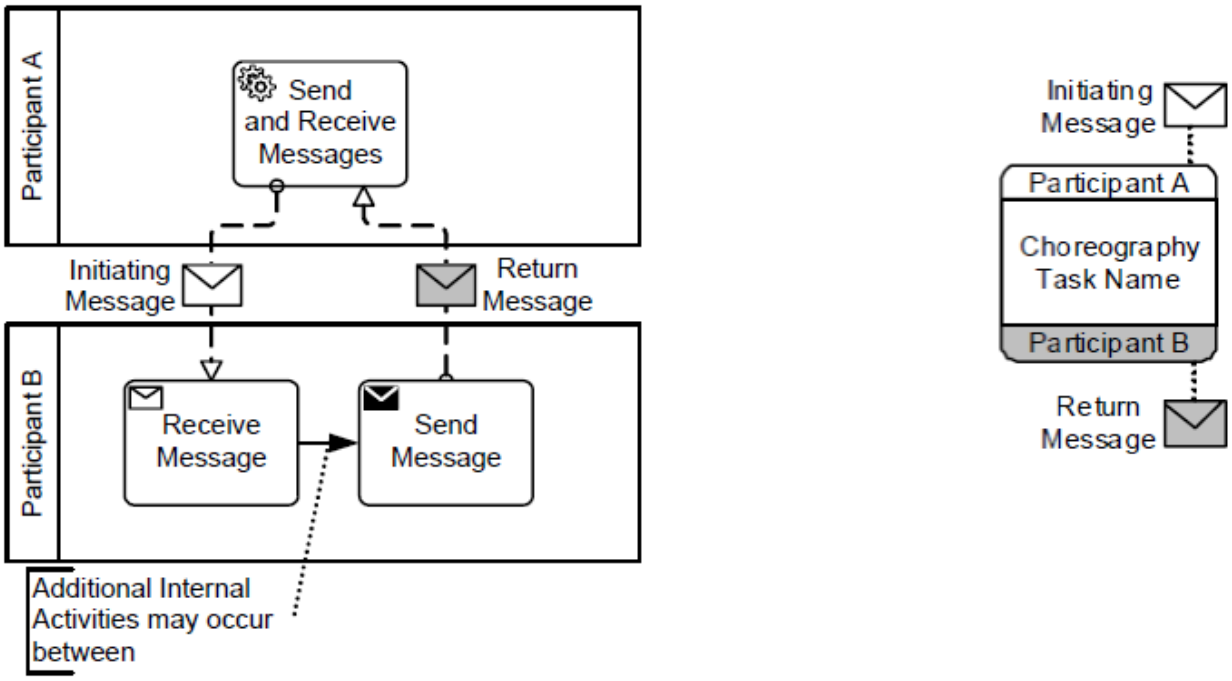

Figura 3.9: *Uma tarefa de coreografia de dois sentidos e seu equivalente em diagrama de colaboração [NF10]*

3.6 Considerações Finais

Neste capítulo se apresentaram aspectos chave em coreografia de serviços Web. Dado que as coreografias para serem realizadas, devem ser mapeadas para processos de negócio executáveis (tais como BPEL), então o *enactment* das coreografias depende do motor de processos executáveis, mas estes não são cientes de coreografia. Em consequência, deveriam existir motores cientes dos aspectos envolvidos nas coreografias de serviços Web. Assim, as atuais implementações de ESB (abordagem utilizada pelos motores de BPEL) não oferecem características especificas das coreografias ([KLN08]), por exemplo, as implementações de ESB não revisam se as mensagens trafegam ou transitam de acordo à descrição da coreografia, isto é, monitoramento. A linguagem *BPEL gold* em [KELL10] visa superar as desvantagens mencionadas, e aborda o problema da necessidade de um ESB ciente de coreografias. WS-CDL+ [KWH07] é outro trabalho que visa ser um motor de

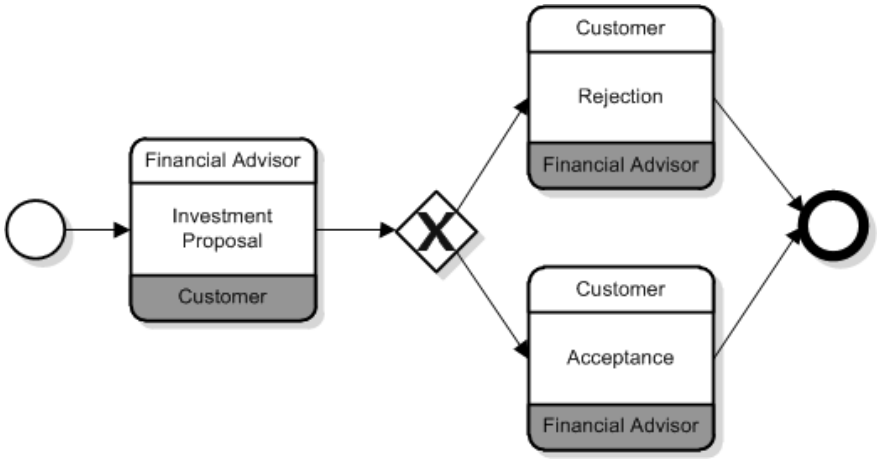

Figura 3.10: *Exemplo de modelo de interação de coreografias, oferta de investimento [KLW10].*

execução de coreografias.

Portanto, já que os atuais motores não são cientes de de coreografias, neste trabalho desenvolveu-se um simulador de coregrafias para possibilitar o *enactment* de coreografias. Assim, o simulador possibilitará a pesquisa em QoS e monitoramento em coreografias, os quais são alvo deste trabalho.

Capítulo 4

Cenários de Coreografias de Serviços

O objetivo deste Capítulo é apresentar exemplos e cenários onde a utilização de coreografias faz sentido. Além disso, apresenta-se também uma comparação entre orquestrações e coreografias com foco em desempenho e dessa maneira conhecer em quais situações aplicar coreografias.

4.1 Oquestração contra Coreografias

O controle centralizado por meio de um mecanismo de orquestração é uma solução válida para cenários encontrados em comércio eletrônico (*e-commerce*), em que quantidades relativamente pequenas de dados intermediários (quando a saída de uma chamada de serviço é diretamente, sem nenhuma alteração, usada como entrada para outro) são movidos entre serviços em um fluxo de trabalho. No entanto, servidores centralizados fazem menos sentido quando se lida com fluxos de trabalho (*workflows*) centrados em dados (GBs/TBs), que são comuns em aplicações científicas. Passando grandes quantidades dados intermédiarios por meio de um motor de orquestração centralizado resulta em uma transferência de dados desnecessária, largura de banda desperdiçada, a sobrecarga do motor de execução e a diminuição do desempenho do fluxo de trabalho [BBRW09].

4.2 Cénarios

Montage

Montage (*Montage Astronomical Image Mosaic Engine*) é um conjunto de ferramentas de software usadas em astrofotografia para montar imagens astronômicas em formato FITS (*Flexible Image Transport System*) em imagens compostas, chamados "mosaicos", que preservam a calibração e fidelidade posicional das imagens originais de entrada.

Montage foi desenvolvido para apoiar a pesquisa científica. Ele permite aos astrônomos criar imagens de regiões do céu que são demasiado grandes para serem produzidos por câmeras astronômicas. Também cria imagens compostas de uma região do céu que foi medido com diferentes comprimentos de onda e com instrumentos diferentes. A imagem composta aparece como se a superfície tivesse sido medida com o mesmo instrumento sobre o mesmo telescópio.

Montagem ilustra várias características de workflows científicos em uso intensivo de dados. Montagem pode resultar em enormes necessidades de fluxo de dados. Os dados intermediários podem ser de três vezes o tamanho dos dados de entrada, por exemplo, um mosaico céu todo pode resultar em 2-8 TB de movimentação de dados. Tal problema pode ser executado diariamente e Montage é essencialmente um grafo acíclico direcionado (DAG).

Montage é usado como cenário em [BBRW09], a fim de demonstrar que as coreografias são melhores do que orquestrações comuns. Para tanto usou-se padrões de fluxo de dados em *workflows*, tais como *fan-in*, *fan-out* e seqüência. O projeto Pegasus[1] é um instrumento de gestão de fluxos de trabalho para construir e habilitar os fluxos de trabalho usados no Montage [JKB+09].

[1] Projeto Pegasus: http://pegasus.isi.edu

Calculando o *redshift*

Este cenário foi tomado dos casos de uso científicos do AstroGrid [BWR09] e envolve a recuperação e a análise de dados em grande escala a partir de vários recursos distribuídos. Um *redshift* (desvio para o vermelho) é um fenômeno ótico ocasionado pelo afastamento da fonte de luz e um caso particular do conhecido efeito Doppler.

O *redshift* fotométrico usa fotometria de banda larga para medir os **redshifts** de galáxias. Enquanto *redshifts* fotométricos têm incertezas maiores que *redshifts* espectroscópicas, eles são a única maneira de determinar as propriedades de grandes amostras de galáxias. Este cenário descreve o processo de consulta a um grupo de bancos de dados distribuídos, contendo imagens astronômicas em diferentes larguras de banda, extraindo objetos de interesse e calcular o *redshift* relativa de cada objeto.

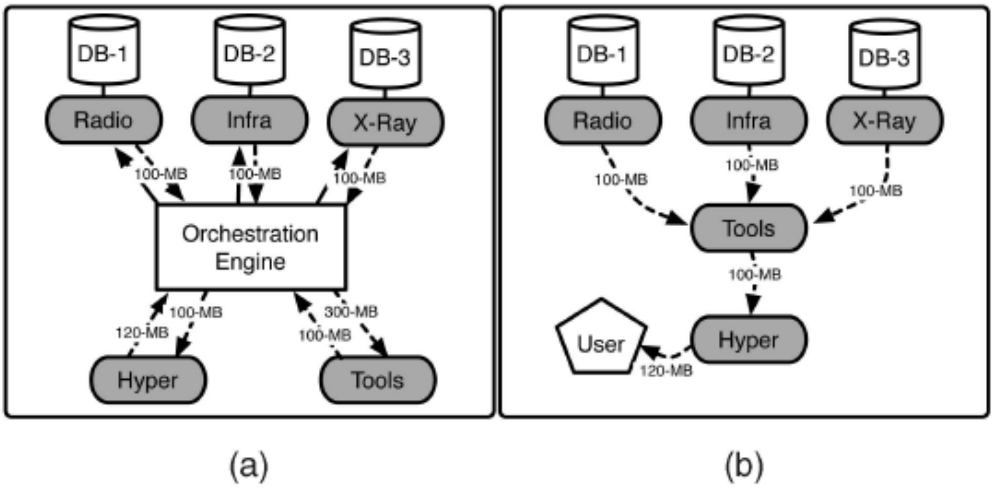

Figura 4.1: *Cenário de redshift do AstroGrid: (a) orquestração, (b) coreografia [BWR09].*

O cenário (ver Figura 4.1) representa um fluxo de trabalho e começa quando um cientista introduz coordenadas de ascensão reta (RA) e de declinação (DEC) no sistema, que definem uma área do céu. Essas coordenadas são usadas como entrada para três bancos de dados remotos astronômicos. Nenhum dos banco de dados possui uma visão completa dos dados requeridos pelo cientista, já que cada banco de dados armazena apenas imagens de uma faixa de uma certa onda. Em cada um dos três bancos de dados, a consulta é usada para extrair todas as imagens dentroMi cuenta es: das coordenadas indicadas que são retornadas ao cientista. As imagens são concatenadas e enviadas para o serviço *SExtractor*, que é uma ferramenta para processamento. *Sextractor* digitaliza cada imagem, por sua vez, e usa um algoritmo para extrair todos os objetos de interesse (posições de estrelas, galáxias, etc) e produz uma tabela para cada uma das bandas, contendo todos os dados. Uma ferramenta de correspondência cruzada (*cross matching*) é, depois, usado para digitalizar todas as imagens e produzir uma tabela contendo dados sobre todos os objetos de interesse no céu em cinco bandas. Esta tabela é então utilizada como entrada para o algoritmo denominado *HyperZ2* que calcula os *redshifts* fotométricos e os anexa a cada valor da tabela usada como entrada. Esta tabela final consiste em arquivos multibanda contendo a posição solicitada, bem como uma tabela contendo para cada fonte todos os parâmetros da saída do *SExtrator* e *HyperZ*, incluindo posições, magnitudes, classificação estelar e *redshifts* fotométricos e intervalos de confiança, a tabela final é retornada para o usuário.

Pegasus

Pegasus (*Planning for Execution in Grids*) é um arcabouço que permite o mapeamento de fluxos de trabalho complexos em recursos distribuídos como a grade. Em particular, Pegasus mapeia um fluxo de trabalho abstrato para uma forma que pode ser executada na rede usando uma variedade

de plataformas computacionais. O arcabouço Pegasus permite aos usuários personalizar o tipo de recurso e seleções de dados desemepenhados, bem como para selecionar as fontes de informação.

Algumas aplicações incluem:

- **Astronomia**: Montage, *Galactic Plane.*
- **Bioinformática**: Sequenciamento de DNA, *Epigenomics*, *SeqWare*, *Association Mapping and Population Genetics in Vervets.*
- **Análise do genoma**: GADU (*The Genome Analysis and Database Update System*).
- **Modelagem do clima**: *Climate ensamble.*
- **Ciência para terremotos**: *CyberShake and Broadband.*
- **Heliosismologia**: *Solar Dynamics Observatory (SDO).*
- *Física*: *Laser Interferometer Gravitational Wave Observatory (LIGO)*, Termodinâmica molecular .
- **Oceanologia**: Predições no oceano.

Assim, as coreografias de serviços podem ser aplicadas nos cenários de fluxo de trabalho mencioandos acima. Já que esses cenários são fluxos de trabalho complexos com uso intensivo de dados, a abordagem da coreografia pode melhorar vários aspectos deles.

Uma rede de valor de negócio

Arfa está visitando ULM (ou algum outro lugar turístico). Ela está gravando filmes e capturando fotos com uma câmera do seu aparelho móvel. O dispositivo móvel tem espaço de armazenamento limitado, mas felizmente ela conhece um serviço Web que pode arquivar, melhorar e hospedar seus filmes online assim que ela complete uma gravação. Ela também está muito animada para compartilhar suas experiências com a sua família e amigos. Portanto, ela quer atualizar alguns blogs com imagens dos lugares e sua descrição histórica. Sua amiga lhe contou sobre um serviço online que pode coletar imagens de seu telefone celular, imprimi-las e enviá-las como cartões postais. Então, ela gostaria de fazer três tarefas: armazenar automaticamente e hospedar seus filmes para armazenamento externo de onde ela e seus amigos podem assistir a qualquer hora, usando seus dispositivos móveis ou estáticos; imprimir automaticamente algumas imagens selecionadas como cartões postais e enviá-los para a sua família e amigos por meio de correio normal; atualizar alguns blogs com imagens e suas descrições históricas.

A coreografia de SLAs resultante deste fluxo de trabalho simples é mostrada na Figura 4.2. Existem dois serviços, chamados de serviços de hospedagem de vídeo e serviço de pós-foto. O serviço de hospedagem de vídeo baixa o vídeo do dispositivo móvel e melhora os arquivos. Qualquer usuário autenticado pode reproduzir o vídeo como no *Youtube.* O serviço pós-foto faz SLAs com dois serviços: o serviço de pós-impressão e o serviço de *E-post.* O serviço *E-Post* é capaz de fazer a sua tarefa por meio da contratação de dois serviços, o *Blog-Service* e o *MMS-Service.* O *Blog-Service* pode atualizar automaticamente os blogs com as imagens e gerar automaticamente histórias sobre a importância histórica com base no seu endereço exato. O *MMS-Service* envia as imagens selecionadas para os amigos em seus telefones móveis.

CDN de fornecimento de conteúdo multimídia

Um tipo particular de serviço que está ganhando interesse em aplicações avançadas é acerca do provisionamento de conteúdo em tempo real. O provisionamento é frequentemente ativado por meio de interações de serviços Web, ou seja, o serviço global combina tecnologias de serviços Web e fornecimento de conteúdo em tempo real, embora o usuário percebe esta composição como um

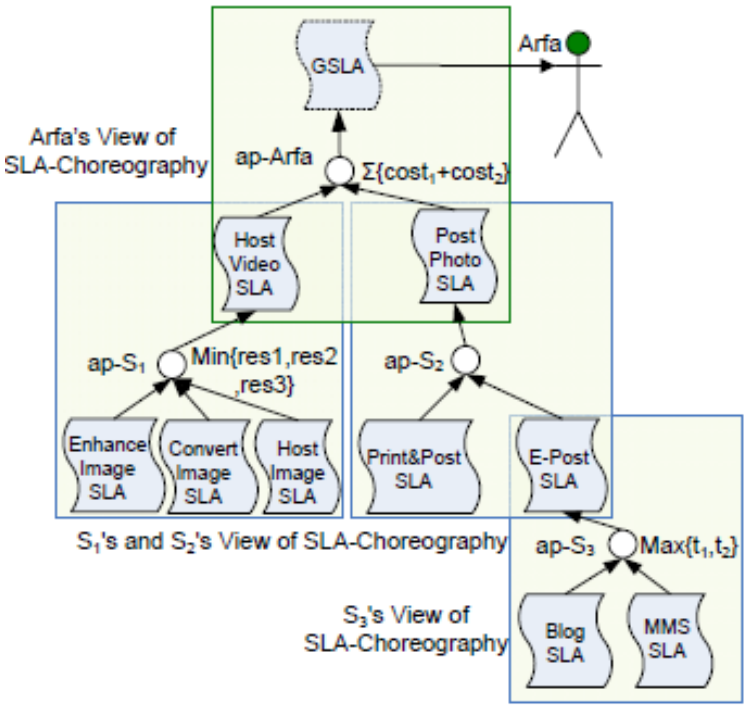

Figura 4.2: *Coreografia de SLA da rede de valor de negócio [UHS11].*

único serviço. Também neste caso, seria importante medir e controlar a QoS de uma abordagem combinada. Eles lidam tanto com a QoS de serviços Web e com o QoS dos objetos entregues, isto é, o conteúdo de *streaming* (imagens, vídeos, texto e som, e assim por diante).

O arcabouço em [BDF+08], lida com os processos interagindo com diferentes atores e oferecendo serviços de valor agregado, os que são capazes de satisfazer as solicitações do usuário para objetos complexos, tais como objetos de *e-learning*, serviços de saúde clínica, ou serviços de *e-governance*. A Figura 4.3 mostra o cenário de referência: um usuário requere, e eventualmente recebe, um serviço complexo gerido por meio de uma coreografia de diferentes serviços Web, um dos quais (WS_3 na figura) controla o provisionamento de conteúdo streaming. WS_1 e WS_2 mostram que vários serviços Web são internamente orquestrados.

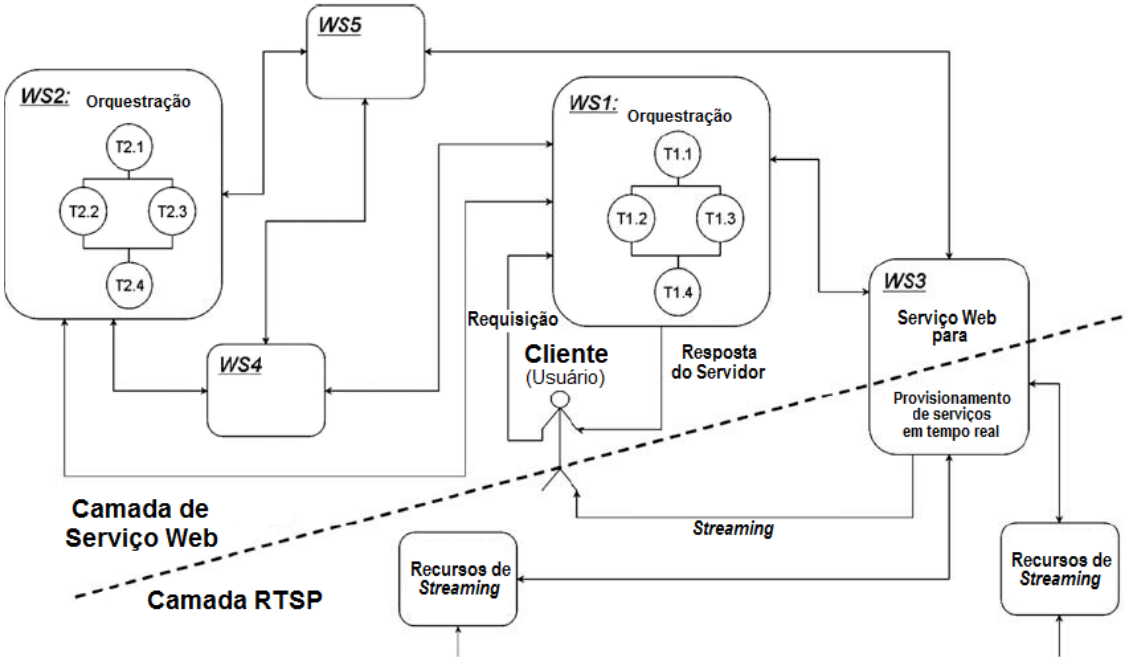

Figura 4.3: *Coreografia de serviços da aplicação de CDN [BDF+ 08]*

Capítulo 5

QoS e Monitoramento em Coreografias de Serviços Web

5.1 Qualidade de Serviço

Um fator chave para habilitar um comportamento adaptativo em sistemas orientados a serviços, e especificamente em composição de serviços, é a disponibilidade de métricas de Qualidade de Serviço (QoS) [Ros09]. O termo QoS surge no escopo das redes de computadores, onde é definido em [CRBS98] como um "conjunto de requisitos de serviços para ser cumprido pela rede enquanto ela realiza o transporte de um fluxo". Na comunidade da SOC, a QoS abrange todos os atributos ou propriedades não funcionais de um serviço, por exemplo, atributos que tem relação com desempenho, confiabilidade, segurança e custos.

QoS tem um papel crucial em sistemas orientados a serviços, por exemplo nas seguintes linhas [MRLD09]: seleção, descoberta, adaptabilidade, monitoramento e composição de serviços ciente de QoS (*QoS-aware*), entre outras. A QoS habilita enlace dinâmico com consciência de QoS a serviços concretos em tempo de execução, habilita também a otimização de serviços compostos, em termos do QoS da composição toda e da adaptação dinâmica de serviços quando acontecerem mudanças [Ros09].

Os atributos de QoS podem ser classificados como determinísticos e não determinísticos [HGDJ08]. Os atributos determinísticos indicam que o valor do atributo é conhecido antes que o serviço seja invocado (p. ex.: custo e segurança), enquanto que nos atributos não determinísticos os valores não são conhecidos antes da invocação do serviço (p. ex.: tempo de resposta e disponibilidade). Tratar com atributos não determinísticos é mais complexo, dado que requerem a realização de cálculos baseados na coleta de dados feita em um monitoramento em tempo de execução. Esses atributos não determinísticos serão o foco neste trabalho, dado que sãos fatores importantes para a realização da adaptação dinâmica em sistemas orientados a serviços [Ros09]. Assim, para os atributos de QoS não determinísticos, as abordagens de monitoramento podem ser usadas para medir continuamente valores de atributos de QoS.

Um atributo de QoS pode ser também estático ou dinâmico [SSDS10]. O valor de um atributo de QoS estático é definido (na especificação ou no projeto), enquanto que o valor de um atributo de QoS dinâmico requer medição e atualização periódica.

A Figura 5.1 exibe uma taxonomia de atributos de QoS que será utilizada como referência nesta pesquisa. O foco do trabalho será em atributos de desempenho como o tempo de resposta, atraso e largura de banda. Para o cálculo e estimativa de valores dos atributos de desempenho serão utilizados os instantes de tempo mostrados na Figura 5.2.

A lista a seguir apresenta uma breve descrição sobre alguns atributos de desempenho de QoS. As variáveis apresentadas nas descrições são apresentadas na Figura 5.2, que representa a linha do tempo da invocação de um serviço genérico.

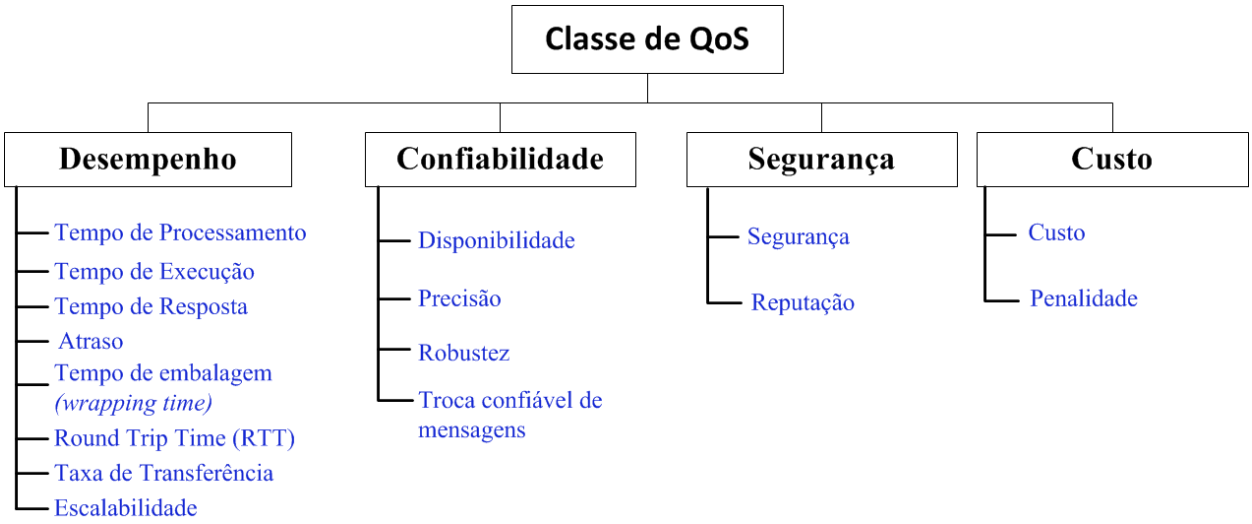

Figura 5.1: *Classificação de atributos de QoS (traduzida de [ADHR05])*

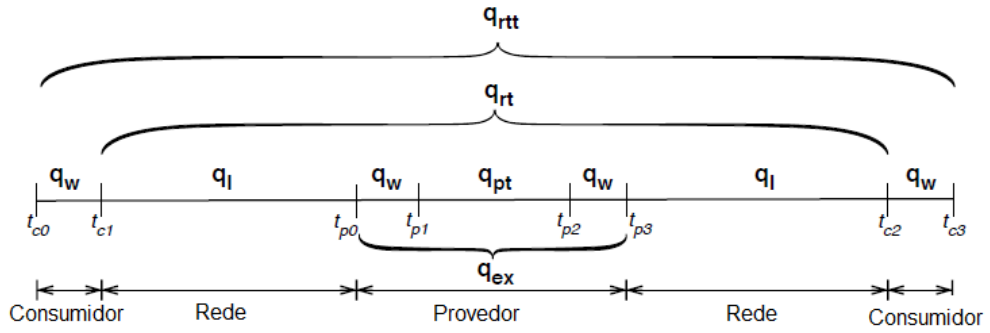

Figura 5.2: *Tempos na invocação de serviços (traduzida de [MRLD09])*

Atributos de Desempenho de QoS

- **Tempo de Processamento**: Tempo necessário para executar uma operação requerida de um serviço. Dado um serviço s e uma operação o, o tempo de processamento é dado por:

$$q_{pt}(s, o) = t_{p2} - t_{p1}$$

- ***Wrapping Time***: É o tempo para encapsular ou desencapsular uma mensagem.
  - No lado do Provedor:
$$q_{wp}(s,o) = t_{p1} - t_{p0} + t_{p3} - t_{p2}$$
  - No lado do Cliente:
$$q_{wp}(c,s,o) = t_{c1} - t_{c0} + t_{c3} - t_{c2}$$
- **Tempo de execução**: Tempo que o provedor de serviço precisa para processar uma requisição.
$$q_{ex}(s,o) = q_{pt} + q_{wp}$$
- **Atraso ou atraso de rede**: Representa o tempo que uma requisição de um serviço demora desde o cliente até o provedor. É influenciado pelo tipo de conexão de rede, roteamento, utilização da rede, largura de banda e tamanho da requisição.
$$q_l(c,s,o) = \frac{t_{p0} - t_{c1} + t_{c2} - t_{p3}}{2}$$
- **Tempo de Resposta:** Tempo necessário para enviar uma mensagem de um cliente c até um fornecedor do serviço s e até que a mensagem de resposta retorne de volta ao cliente.
$$q_{rt}(c,s,o) = q_{ex}(s,o) + 2 * q_l(s,o)$$
- ***Round Trip Time***: Tempo total que é consumido, isto é, tempo de resposta mais o *wrapping time*.
$$q_{rtt}(c,s,o) = q_{wc}(s,o) + q_{rt}(s,o)$$
- **Taxa de Transferência**: Número de requisições r por operação o que foram processadas por s e retornadas ao cliente c com sucesso dentro de um intervalo $[t_0, t_1]$.
$$q_{tp}(c,s,o) = \frac{r}{t_1 - t_0}$$
  Depende principalmente do poder do *hardware* e da largura de banda do fornecedor de serviço.

Atributos de Confiabilidade de QoS

- **Disponibilidade**: Define a probabilidade de se um serviço s está apto e rodando para produzir resultados corretos dentro de um intervalo $[t_0, t_1]$.
$$q_{av}(s) = 1 - \frac{t_d}{t_1 - t_0}$$
- **Precisão**: É a taxa de sucesso produzido pelo serviço s.
$$q_{ac}(s) = 1 - \frac{r_f}{r_t}$$
- **Robustez**: Probabilidade que um sistema possa reagir adequadamente a mensagens de entrada inválidas, incompletas ou conflitantes.
$$q_{ro}(s) = \frac{1}{r_t} \sum_{i=1}^{n} f(resp_i(s))$$
  Onde $resp_i(s)$ é uma função auxiliar, que é a $i - sima$ reposta produzida por um serviço s. n é o numero total de requisições para s. $f(m)$ representa se a resposta de uma mensagem m é

válida ou inválida para uma entrada dada, é calculada com a seguinte fórmula.

$$f(m) = \begin{cases} 1 : isValid(m) \\ 0 : \neg isValid(m) \end{cases}$$

5.1.1 Topologia de atributos e métricas de QoS

Além das taxonomias mencionadas, os atributos de qualidade e métricas também podem ser classificados de diferentes perspectivas.

A partir de uma perspectiva de domínio. Os atributos de qualidade são classificados como atributos independentes de domínio ou específicos de domínio [Hil09]:

- **Atributos independentes de domínio**: São aqueles que podem ser aplicados a qualquer tipo de serviço Web. Por exemplo, o tempo de resposta, a disponibilidade e o custo.
- **Atributos específicos de domínio**: São aqueles que podem ser aplicados apenas a um determinado ou determinados domínios. Por exemplo, em um domínio de previsão de tempo, um atributo de qualidade poderia ser a precisão da previsão.

De uma perspectiva mensurável, as métricas de qualidade podem ser classificados de acordo com o método em que os dados podem ser obtidos. Nesse sentido, três categorias foram identificadas [Hil09]:

- **Métricas anunciadas pelo Provedor**: Métricas em que os seus valores são obtidos pelo anúncio do provedor. Um claro exemplo seria o custo do serviço.
- **Métricas classificadas pelo Consumidor**: Métricas em que seus valores são dados a partir das opiniões de usuários. Por exemplo, a reputação do serviço obtido a partir de uma média de opiniões dos clientes.
- **Métricas Observáveis**: Métricas em que os seus valores são obtidos mediante monitoramento ou testes. Um claro exemplo é o tempo de resposta ou a disponibilidade.

Outro tipo de classificação é a partir do fato de que algumas métricas de qualidade podem ser obtida como uma função de agregação de outras métricas de qualidade. Por exemplo, o tempo de resposta médio é a média de um conjunto de tempos de resposta atuais em diferentes intervalos de tempo. Essas métricas de qualidade são nomeadas métricas derivadas ou calculadas, enquanto os outros são nomeados métricas básicas ou atômicas.

5.2 QoS em Sistemas Orientados a Serviços

Na computação orientada a serviços (SOC), o fornecimento de serviços com garantia de qualidade exige mecanismos que incluam modelos de qualidade de serviço (QoS) como característica chave. De fato, os modelos de QoS fornecem uma base adequada para o cumprimento de QoS em ambientes orientados a serviços [MGI09].

Em tais ambientes, a garantia de QoS pode ser afetada por diversos fatores incluindo o hardware, a infraestrutura de rede, o nível de qualidade oferecido pelos serviços de aplicação e pelas características do usuário final (mobilidade por exemplo). Isso implica que, a fim de se obter uma avaliação exata da QoS, nenhum destes aspectos devem ser negligenciados na fase de modelagem de QoS. Parâmetros que devem ser considerados em um modelo de QoS para coreografias são similares àqueles considerados em sistemas P2P [MGI09]: (i) o ambiente do serviço e o hardware e infraestrutura de rede subjacente, (ii) os serviços disponíveis, e (iii) os usuários finais.

Diferentes métodos e ferramentas para capturar e analisar o desempenho de serviços Web tem sido desenvolvidos. Em geral as abordagens para a avaliação da capacidade dos serviços Web diferem

pelo conjunto de métricas de QoS definidas em tempo de projeto ou em tempo de execução. Isto pode dificultar a escolha de uma abordagem para a avaliação de QoS de uma aplicação.

No contexto de uma coreografia de serviços Web, é proposto em [Ros09] um modelo de QoS multi-camada que permite integrar coerentemente informação de QoS em vários níveis de abstração (Figura 5.3): a camada de coreografia, a camada de orquestração e a camada de serviços. A seguir é apresentada uma descrição resumida de cada camada desse modelo.

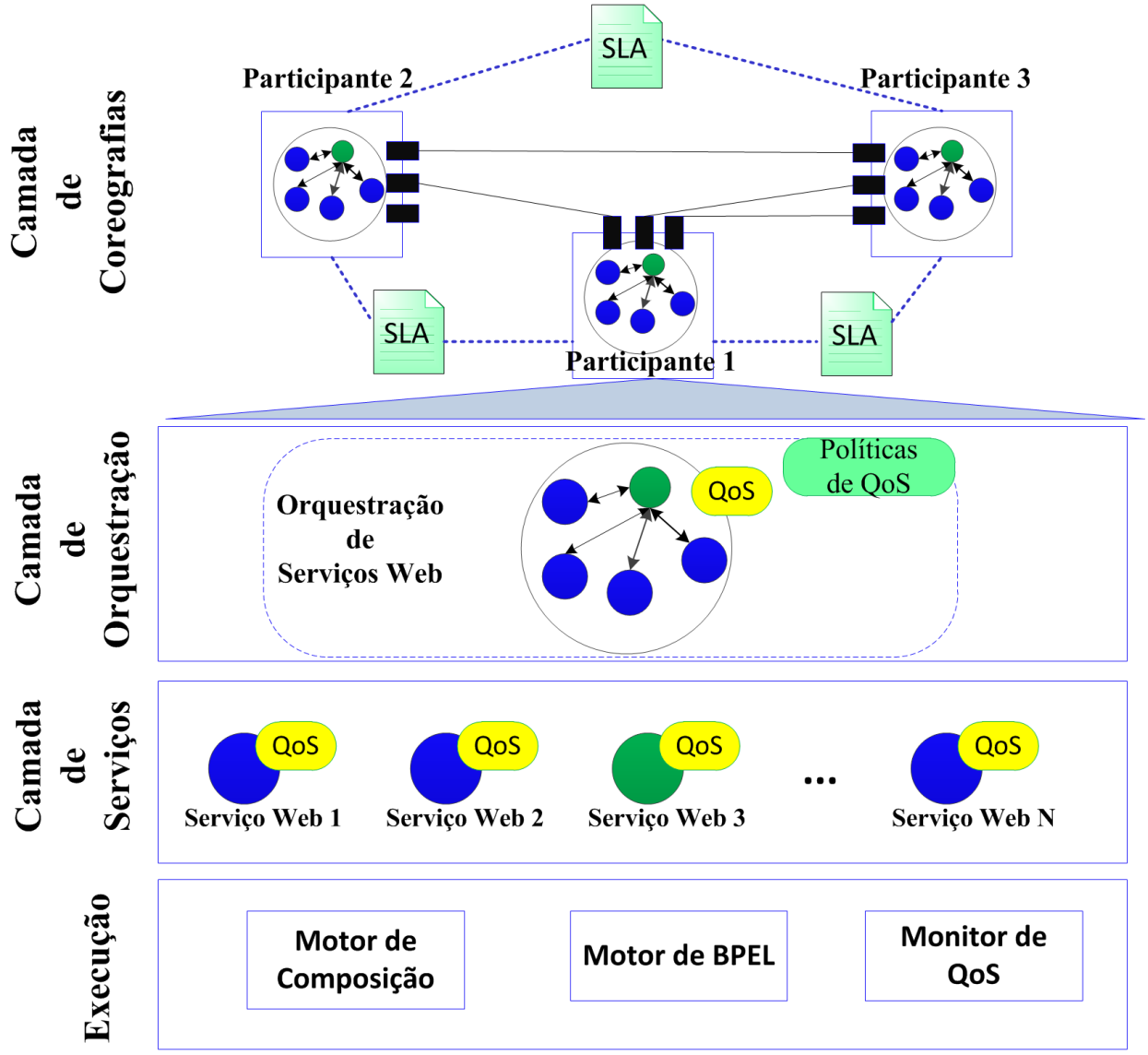

Figura 5.3: *Modelo multi-camada de QoS para coreografias de serviços Web (baseada em [Ros09])*

Camada de Serviços

Os serviços em sistemas orientados a serviços, podem ser invocados por usuários que na maioria dos casos são desconhecidos em tempo de projeto e implantação. Por isso, é importante que um serviço forneça uma descrição dos aspectos não funcionais, além dos funcionais (por exemplo, em um documento WSDL). Esta camada representa todos os serviços atômicos dentro de um sistema orientado a serviços. Nesta camada são definidos os atributos de QoS tais como tempo de resposta, disponibilidade, largura de banda, etc.

Os atributos definidos nesta camada serão utilizados nas camadas superiores (orquestração ou coreografia) para, através de cálculos de agregação de QoS, obter o QoS global e desta maneira definir políticas de QoS na orquestração e SLA na coreografia.

A Figura 5.4 descreve os componentes de um serviço Web usando um diagrama de classes UML. Esse componentes incluem o nome do serviço Web e suas mensagens, operações, *port*, *portType* e *bindings* associados. Os atributos de qualidade de serviço associados com o serviço, operação do serviço e as mensagens são resumidas na Tabela 5.1.

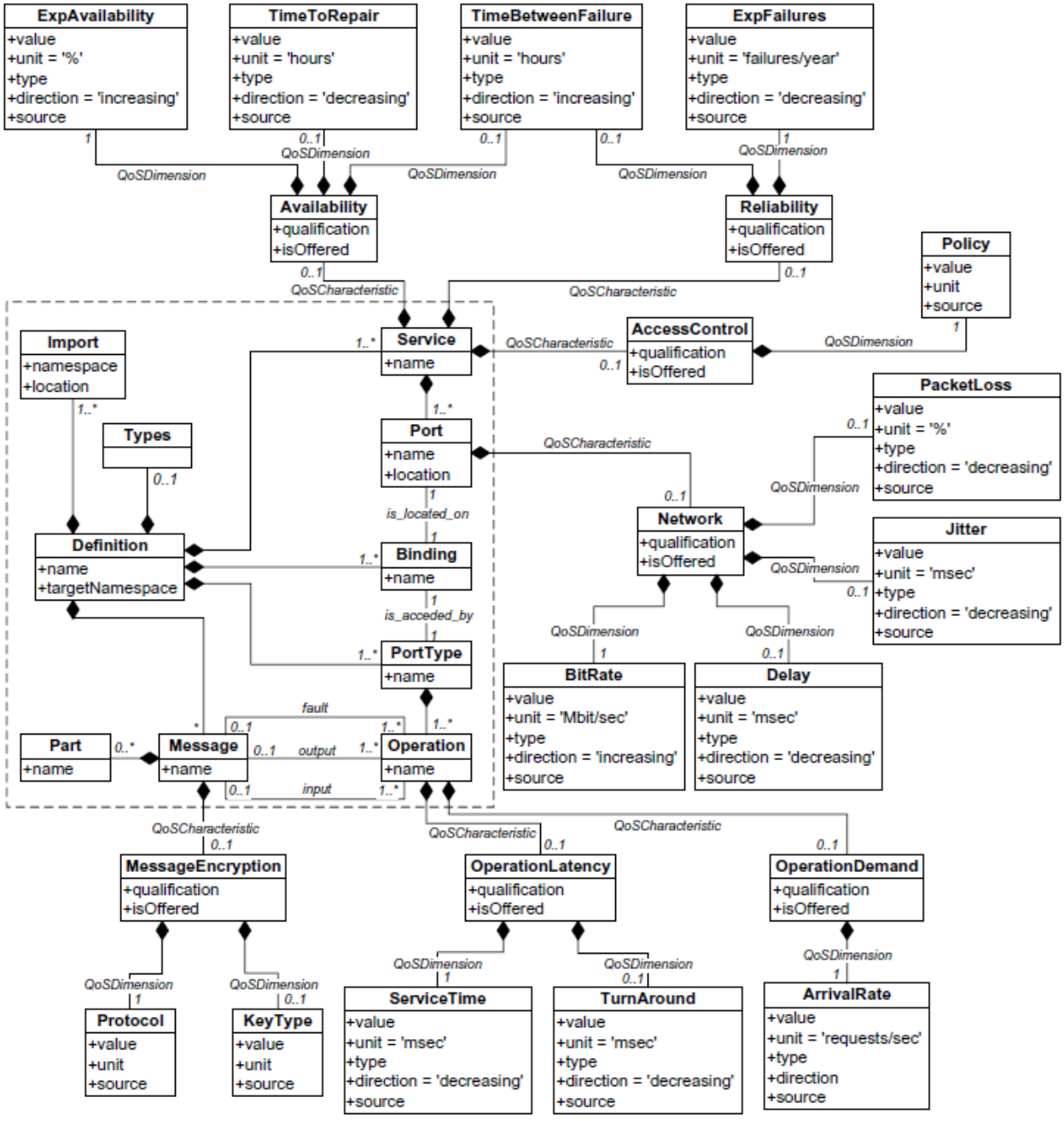

Figura 5.4: *QoS habilitada no modelo de serviços Web de acordo com WSDL [Pan10]*

Nesse modelo, os atributos de qualidade no componente serviço são a confiabilidade e a disponibilidade. A confiabilidade está baseada no número esperado de falhas ao longo de um intervalo de tempo e tempo entre falhas. O atributo de disponibilidade é uma agregação de disponibilidade esperada, tempo para a reparação e o tempo entre a falha. O tempo de reparo é o tempo necessário para reparar o serviço após uma falha. A disponibilidade esperada pode ser calculada como a razão entre o tempo de atividade e o tempo total (que inclui o tempo de *uptime* e *down time*). O atributo de qualidade no na "operação" inclui a sua demanda e latência. A demanda pode ser computada em termos da taxa de chegada da requisição respectiva para essa operação. A latência pode ser especificada em termos do tempo de serviço e o tempo de ida e volta. O atributo de qualidade para

"mensagem" depende do protocolo utilizado para uma saída segura e entrada (protocolo) e tipo de chaves utilizadas para criptografar mensagens.

Tabela 5.1: *QoS no modelo WSDL estendido*

QoS em um serviço Web

Tipo	Parâmetro de QoS	Campo
Serviço	Disponibilidade	TTR TBF ExpA
	Confiabilidade	TBF Exp *Failure*
Operação	Latência	Tempo de serviço TBF
	Demanda	TBF Exp *Failure*
Mensagem	Protocolo	Enviar/Receber
	Tipo de Chave	Enviar/Receber

Camada de Orquestração

Nesta camada as especificações dos SLA's são mapeadas para políticas concretas de QoS para cumprir o SLA no correspondente processo de negócio do participante [Ros09]. WS-QoSPolicy [REM+07], que é uma extensão de WS-Policy[1], pode ser usado para especificar as políticas de QoS.

WS-Policy é um arcabouço extensível definido na pilha de serviços Web, que é uma coleção de políticas em forma de asserções. Tais asserções de políticas definem requisitos, capacidades ou outras propriedades de um determinado objeto, por exemplo, mensagens, *endpoints*, operações, entre outros.

WS-QoSPolicy define um modelo de asserções para atributos de QoS na camada de serviços. Com essas asserções de políticas QoS, requisitos de QoS para serviços podem ser integrados em processos de negócio, cabendo ao motor de processos tratar essas asserções apropriadamente, por exemplo, disparando eventos para algum serviço ou entidade que tome as medidas necessárias quando um valor específico for violado.

Para avaliar e validar os valores no SLA, precisa-se de informação de QoS de cada serviço através de monitoramento, levando em consideração que os valores dos atributos de QoS medidos devem ser agregados para calcular a QoS da composição. A agregação de atributos de QoS é realizada por meio de um cálculo incremental baseado em padrões de fluxo de trabalho (*workflows*) bem definidos [Van03]. Para cada um desses padrões uma regra de agregação de QoS tem que ser aplicada para obter a QoS do padrão. Aplicando este processo recursivamente consegue-se a QoS global da composição.

Camada de Coreografias

É o topo do modelo de QoS multi-camada e tem como principal meta expressar as garantias e obrigações em termos de QoS entre as partes contratuais em um alto nível de abstração [Ros09]. O acordo contratual entre dois participantes é geralmente descrito como SLA (Acordo de Nível de Serviço) [JMS02]. O SLA é importante quando se integra serviços externos dentro de processos de negócio (coreografias, orquestrações ou *workflows*) [MRLD09]. Um SLA fornece um conjunto de operações, funções e predicados para definir restrições ou condições de QoS. Assim, um SLA define principalmente as seguintes três seções:

[1]WS-Policy: recomendação pela W3C, http://www.w3.org/TR/ws-policy/

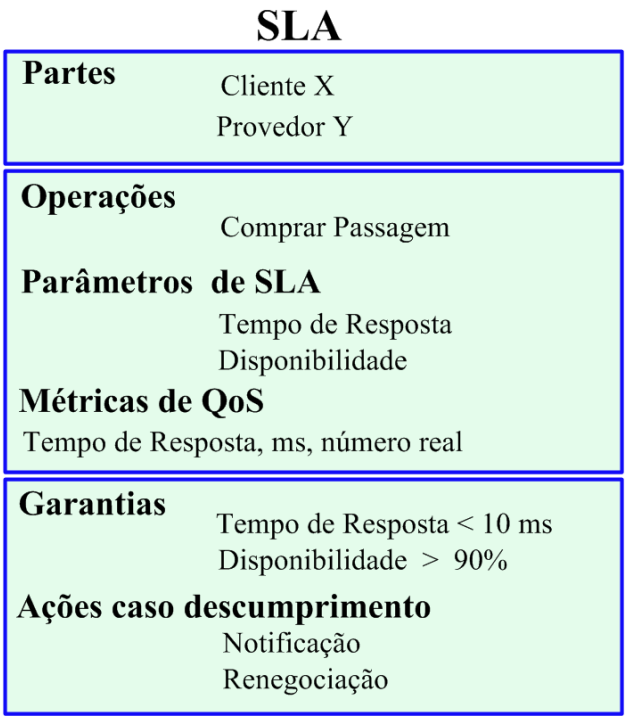

Figura 5.5: *Exemplo de SLA [KL03]*

- **Participantes**: Identifica todas as partes contratuais, incluindo a identificação e todas as propriedades técnicas, tais como as descrições da interface ou o *endpoint* do serviço.
- **Descrições de serviços**: São definidas todas as características do serviço: as operações envolvidas e os parâmetros de SLA.
- **Obrigações**: Define as restrições para garantir valores nos parâmetros de SLA. São representadas e especificadas através de SLOs (*Service Level Objectives*) usando parâmetros de SLA. A listagem 5.1 mostra a definição de dois parâmetros de SLA: tempo de execução e a taxa de transferência. Para cada parâmetro se definem o nome, o tipo, a unidade de medida e a métrica de QoS.

```
1    <wsp:Policy>
2      <wsp:All>
3  <qosp:ExecutionTimeAssertionunit="msec" predicate="Less" value="1500"/>
4  <qosp:ThroughputAssertionunit="ops" predicate="GreaterEqual" value="130"/>
5      </wsp:All>
6    </wsp:Policy>
```

Listagem 5.1: *Exemplo de parâmetro de SLA [KL03]*

Toda vez que uma restrição é violada pelo provedor de serviços, eventos são disparados para notificar a violação. Após a notificação da violação podem ser tomadas ações como punições, adaptação ou mudança do provedor de serviços [MRLD09]. No lado do provedor, uma notificação de violação pode servir para tomar ações corretivas ou simplesmente selecionar um outro serviço com as mesmas funcionalidades e realizar uma substituição dinâmica. A Figura 5.5 apresenta um exemplo de um SLA e os elementos que a compõem.

A definição de SLA e QoS joga uma regra crucial em processos de negócios inter-organizacionais. Cada participante oferece serviços a outros particapntes sobre a Internet, dos quais os últimos precisam executar seus negócios [REM+07]. Por isso, um certo grau de confiabilidade respeito do tempo de resposta, vazão, *uptime*, entre outros, é desejado e tem de ser especificado e explicitamente expresso nas etapas iniciais da fase de modelagem. A Figura 5.6 ilustra uma coreografia com três

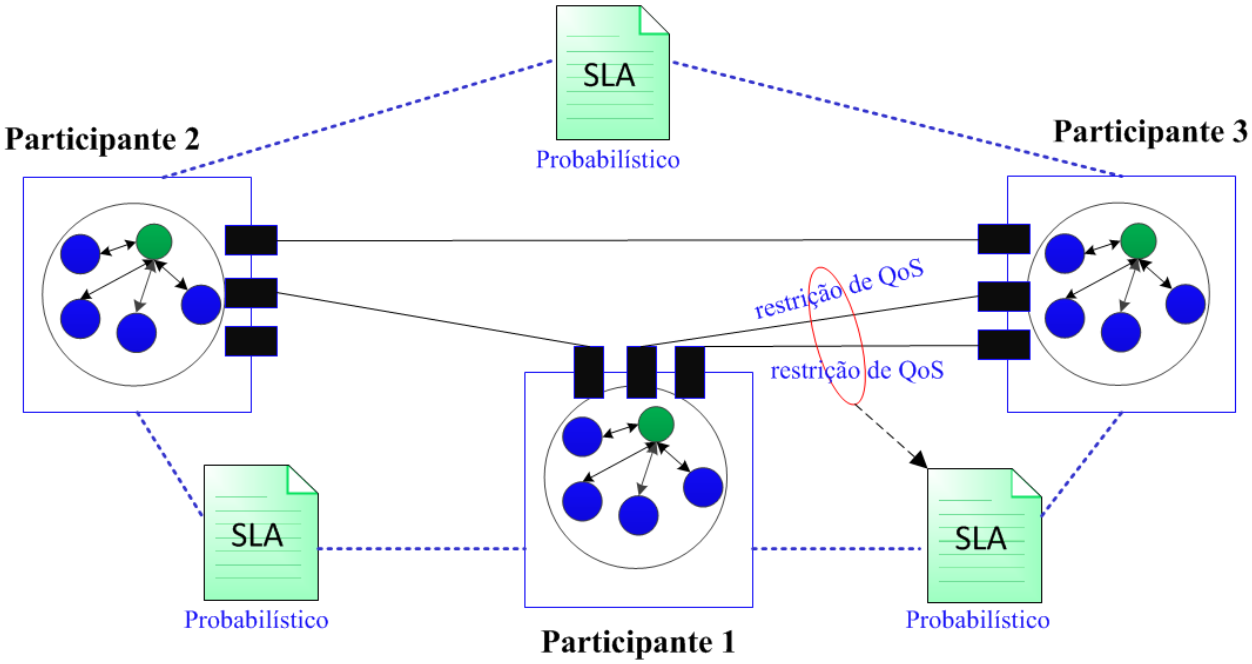

Figura 5.6: *Definição de SLAs na camada de coreografias de serviços*

participantes e seus respectivos relacionamentos. Para cada relacionamento (interação entre serviços) um SLA é definido entre os participantes para regular o grau que os participantes precisam para cumprir seus objetivos de negócio.

Dado que uma coreografia de serviços é um descrição desde um ponto de vista global das interações do seus participantes, não pode ser executada direitamente porque não envolve o aspecto interno de um participante. Assim, o SLA precisa ser cumprido também no aspecto interno (comportamento não visível) de um participante. Em consequência, uma descrição SLA precisa ser mapeada em politicas concretas de QoS na camada de Orquestração.

5.3 SLAs Probabilísticos

Tipicamente, as restrições de QoS são estabelecidas de maneira rígida (*hard contracts*), isto é, com limites absolutos [RBHJ08]. Por exemplo, o tempo de resposta de um serviço Web tem de ser menor que $10ms$. Porém, o comportamento dinâmico de parâmetros de QoS, como o tempo de resposta, dificulta as suas representações por restrições rígidas. A Figura 5.7 mostra um histograma da distribuição de probabilidade dos tempos de resposta medidos para um serviço [RBHJ08], o que serve para ilustrar o comportamento dinâmico do tempo de resposta.

Assim, uma melhor maneira de representar o comportamento dinâmico dos parâmetro de QoS é através de restrições não rígidas, as quais não utilizam valores ou limites absolutos. Um exemplo de restrição não rígida é requisitar que o tempo de resposta seja menor do que $10ms$, em 95% dos casos.

A restrição anterior ainda não é suficiente para refletir a dinâmica dos parâmetros de QoS. A solução é utilizar restrições não rígidas de QoS baseadas em distribuições de probabilidade, isto é, contratos probabilísticos [ZYZ10]. Uma restrição probabilística é da forma: "Para o parâmetro tempo de resposta, eu, provedor de serviço forneço a distribuição de probabilidade, e garanto que ela será respeitada".

Na prática essas restrições probabilísticas são definidas como um conjunto de quantis dos parâmetros de QoS. Esses quantis definem uma distribuição empírica de probabilidade de parâmetros de QoS. Por exemplo, a Tabela 5.2 mostra que quantis de 90%, 95% e 98% correspondem a tempos de resposta de $6494ms$, $13794ms$ e $23506ms$ respectivamente. Assim, o quantil 95% expressa que em 95% dos casos o tempo de resposta é no máximo $13794ms$.

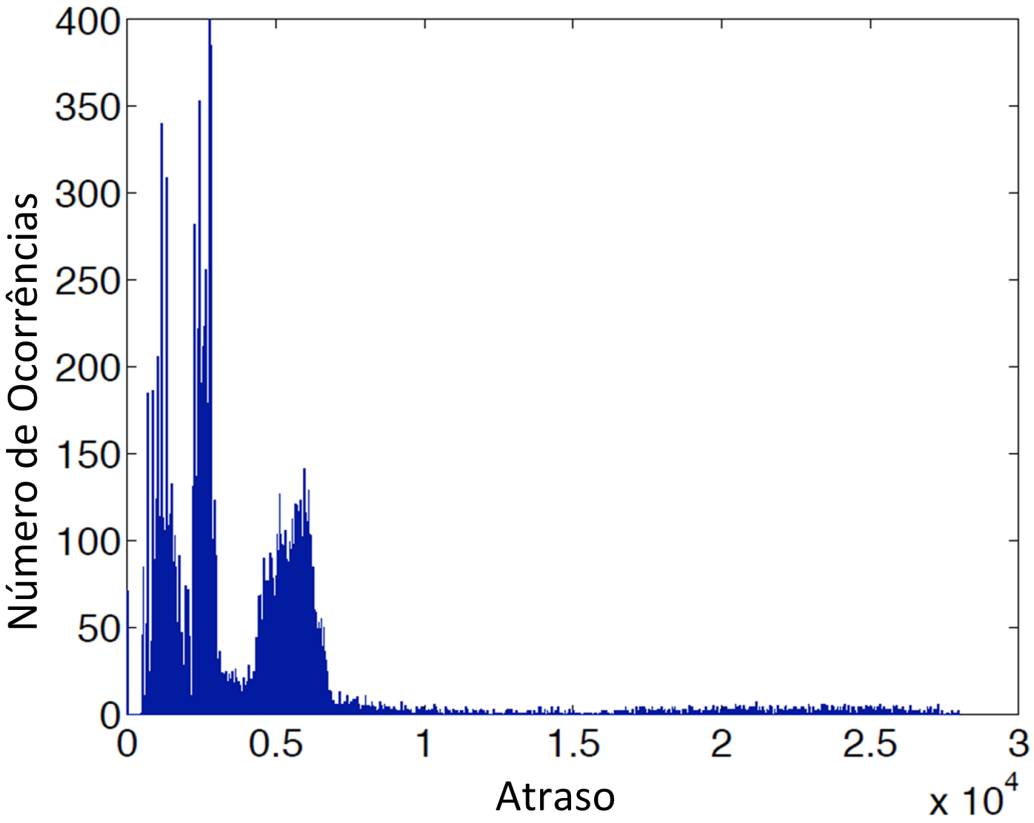

Figura 5.7: *Registro de medidas do tempo de resposta para um serviço Web (traduzida de [RBHJ08])*

Tabela 5.2: *Quantis de tempos de resposta (obtida de [RBHJ08])*

Quantis	Tempo de Resposta
90%	6494 ms
95%	13794 ms
98%	23506 ms

Uma vez calculadas as distribuições de probabilidade dos parâmetros de QoS, estes precisam ser agregados segundo a composição. Atualmente existem poucas pesquisas acerca de restrições probabilísticas de QoS em serviços Web [HWTS07], [RBHJ08], [RBJ09] e [ZYZ10]. Não há, entretanto, abordagens específicas para coreografias de serviços Web.

5.4 Trabalhos Relacionados: QoS em Coreografias de Serviços Web

Existem várias abordagens para integrar modelos de QoS em composição de serviços, mas poucos focados em coreografias de serviços. WS-CDL não tem integrado QoS apesar que tem demandado uma especificação de QoS no seus requerimentos [ABPRT04]. Em [XCHZ07], propõe-se um modelo formal que incorpora informação de QoS, estendendo a pequena linguagem *Chor*[YZQ+06] (que é um WS-CDL simplificado) para adicionar informação QoS, sobretudo tempo e custo, em um nível sintático e semântico para obter o QoS global da coreografia.

Em [PC08], propõe-se um modelo formal para estimar o minimo, máximo e a média do costo(atributo não funcional) de uma coreografia. Tal modelo proposto adiciona características semânticas para incorporar o atributo costo no seu modelo, que é uma extensão do WS-CDL. Este modelo associa o costo com instâncias de um papel(*role instance*) em lugar de *role type* do WS-CDL. Em [XCZH09], propõe-se uma abordagem nova para predizer o QoS de uma coreografia de serviços especificada

em WSCI. Utiliza uma Rede de Petri Estocástica Generalizada (*GSPN* [2]) como modelo intermediário. A partir do modelo GSPN obtido do mapeamento do modelo WSCI, avaliações analíticas da árvore de métricas QoS são realizadas. Em [Pan10], propõe um modelo multidimensional de QoS para coreografias. Tal QoS de coreografia é definido em termos do papel dominante, relacionamento dominante e interação dominante. Essas características são avaliadas baseadas no QoS dos papéis, operações e mensagens. Sua proposta é desenvolvida como uma extensão do metamodelo de WS-CDL para suportar QoS.

Somente em [Ros09], propõe-se um modelo multicamada para integrar QoS e SLA no nível de coreografia, orquestração e serviços. No nível de coreografias integra WSLA na linguagem WS-CDL para especificar o acordo do nível de serviço (SLA) entre os participantes. E para realizar seu cumprimento internamente em cada participante realiza o mapeamento a políticas QoS na camada de orquestração, que por sua vez utiliza os atributos QoS no nível de serviços. Até hoje, nenhum propõe um modelo de QoS para integrá-lo em BPMN, que é o padrão para modelar processos de negócios colaborativos, incluindo coreografias.

5.5 Monitoramento em Coreografias de Serviços Web

A eficiência e automatização do monitoramento de atributos de QoS em serviços Web é fundamental para linhas de pesquisa como a seleção, composição e otimização de serviços Web com consciência de QoS no escopo da SOC [MRLD09], [JTK00]. Atualmente os serviços Web não fornecem informação de QoS como parte da sua interface de contrato ou funcionalidade, por exemplo, um documento WSDL.

Alguns valores de atributos de QoS podem ser obtidos tanto em um monitoramento no lado do cliente quanto no lado do provedor, sendo que a depender do atributo, a sua medição pode ser mais precisa em um ponto específico da comunicação [MRLD09]. Por exemplo, no lado do provedor a medição do tempo de execução é mais precisa do que no lado cliente, já que no provedor é conhecida a implementação do serviço. Nesse caso a abordagem é dita intrusiva pelo fato de ser necessário realizá-la no provedor ou conhecer a implementação no lado do provedor.

Uma abordagem de monitoramento do lado do cliente é dita não intrusiva, dado que não há necessidade de se conhecer o comportamento interno do serviço fornecedor [HGDJ08]. Atualmente existem muitas abordagens para a medição de atributos de QoS. A lista a seguir apresenta uma classificação de algumas das abordagens existentes.

- **Instrumentação do Lado do Provedor**: Neste caso o monitoramento é realizado do lado do provedor de serviços. Existem dois tipos de instrumentação, a intrusiva e a não intrusiva. A primeira é realizada através de modificações no código-fonte do serviço. A segunda não necessita de acesso ao código-fonte do serviço. Para tanto, mecanismos como aqueles disponibilizados pelo sistema operacional do computador onde o serviço reside são utilizados. Esta abordagem não suporta a medição de atributos como o atraso de comunicação e em consequência os que dependem dele. Além disso o consumidor do serviço tem que confiar nas medidas do provedor.

- **Intermediários de SOAP**: Esses intermediários interceptam, processam e direcionam mensagens SOAP. Nesse ambiente, o monitor fica localizado entre o consumidor e o provedor de serviços para interceptar as mensagens e medir os atributos de QoS. Na prática o monitoramento pode ser realizado no lado cliente, no lado provedor ou por um parte externa e confiável para ambos os lados. Dado que atua como um *proxy* tratando todas as requisições para realizar medições de QoS, essa entidade pode se tornar um gargalo no sistema. Além disso, podem ser necessárias informações adicionais nas mensagens SOAP para calcular atributos de QoS [RR09].

[2] GSPN: Generalized Stochastic Petri Net

- ***Probing***: Consiste em enviar requisições de teste ao fornecedor do serviço em tempos regulares. Não precisa interceptar mensagens, e o monitor pode estar no cliente, no provedor ou entre os dois. O fator chave é o intervalo de tempo entre cada requisição de teste ao fornecedor, já que isso afeta a precisão da medição. Esta abordagem é utilizada principalmente para medir a disponibilidade de um serviço e o tempo que ele está ativo.

- ***Sniffing***: Utiliza técnicas de análise dos pacotes de rede, por exemplo, na invocação de um serviço, permitindo desta maneira, a medição de atributos como o atraso e o tempo de resposta.

5.6 Trabalhos Relacionados

A pesquisa apresentada neste documento propõe uma técnica para realizar monitoramento de coreografias de serviços Web utilizando SLAs probabilísticos, cujas restrições são especificadas por meio de quantis das distribuições de probabilidades dos parâmetros de QoS dos serviços Web. Além disso, o monitoramento que se pretende desenvolver é não intrusivo. Assim, são resumidos a seguir alguns trabalhos relacionados e organizados em duas subseções: Monitoramento baseado em QoS de coreografias de serviços Web e Monitoramento de serviços usando SLAs probabilísticos.

5.6.1 Monitoramento de Coreografias de Serviços Web Baseado em QoS

Existem várias abordagens para integrar modelos de QoS em composição de serviços, mas poucos focados em coreografias de serviços. Atualmente, as linguagens de coreografia não possuem um modelo de QoS na sua especificação, embora WS-CDL tenha demandado uma especificação de QoS nos seus requisitos [ABPRT04]. [XCHZ07], [PC08] e [Pan10] propõem modelos formais de coreografia de serviços Web para incorporar informação de QoS. Estas abordagens estão focadas no escopo da especificação de coreografias e levam em consideração apenas o tempo de resposta. Diferente das propostas em [XCHZ07], [PC08] e [Pan10], a proposta de pesquisa apresentada neste documento, pretende realizar monitoramento de coreografias além do escopo de especificação e serão levados em consideração outros atributos de QoS como o atraso e a largura de banda, além do tempo de resposta.

Em [WKK+10], apresenta-se uma abordagem baseada em eventos para monitoramento de processos inter-organizacionais que compõem coreografias de serviços Web. Esse trabalho está focado em realizar acordos entre os monitores dos participantes da coreografia e realizar trocas de dados de monitoramento para realizar um monitoramento global. A abordagem proposta habilita o rastreamento e avaliação de métricas de processos, que neste contexto são chamados de KPI (Indicadores Chave de Desempenho). Em síntese, desenvolve-se uma infraestrutura de monitoramento de coreografias baseada em eventos e o foco não são os atributos de QoS, apesar de que trata KPIs que são de mais alto nível do que QoS. Em [ML08], é proposta uma abordagem similar. Os autores propõem uma abordagem de monitoramento de processos de negócio entre múltiplas organizações. Diferentemente dessas abordagens, esta proposta de dissertação foca na detecção de violações de restrições de QoS por meio de SLAs, e o monitoramento das coreografias não abrange restrições de mais alto nível como o uso de KPIs, os quais estão mais associados no contexto de processos de negócio.

Em [UPSB10] é proposta uma abordagem baseada em agregação de SLAs para garantir o cumprimento dos requisitos de QoS em coreografias de serviços Web A proposta está focada na arquitetura de um arcabouço de validação baseado em regras. O trabalho em [UPSB10] não possui foco em QoS, já que o foco é fornecer uma infraestrutura baseada em regras de validação para especificar SLAs hierárquicos para interações de grande complexidade em uma coreografia. Porém, apesar de não propor uma abordagem de monitoramento de QoS como na presente pesquisa, o avanço na definição de SLAs para interações complexas em uma coreografia é útil para os objetivos da dissertação.

Em [XCZH09], propõe-se uma abordagem para predizer a QoS de uma coreografia de serviços especificada em WSCI. É utilizada uma Rede de Petri Estocástica Generalizada (*GSPN*[3]) como modelo intermediário. A partir do modelo GSPN obtido do mapeamento do modelo WSCI, avaliações analíticas da árvore de métricas QoS são realizadas. São definidas regras de transformação para mapear atividades, padrões de encaminhamento (*sequence*, *all*, *choice*, *switch*, *while*) e outras construções para fragmentos ou partes do GSPN. A partir da matriz de transições probabilísticas derivada do GSPN, métodos analíticos são utilizados para avaliar a árvore de métricas de QoS, as quais estão relacionadas com o tempo de resposta. Os autores [XCZH09] utilizam o método de Monte-Carlo para validar os resultados teóricos. Um trabalho do [XCZH09] é a utilização da linguagem WSCI, que foi oficialmente descontinuada. Além disso o foco está no escopo da especificação da coreografia e apenas a métrica de tempo de resposta é utilizada. Na pesquisa proposta neste documento, utiliza-se o GSPN com suporte de QoS para avaliar o desempenho de coreografias de modo a definir de requisitos de QoS. Além disso, leva-se em conta outros atributos de QoS além do tempo de resposta.

Em [Ros09], propõe-se um modelo multi-camada para integrar QoS e SLA no nível de coreografia, de orquestração e de serviços. No nível de coreografias integra WSLA na linguagem WS-CDL para especificar SLAs entre os participantes. Para realizar seu cumprimento internamente em cada participante realiza-se o mapeamento de SLA para políticas de QoS na camada de orquestração, que por sua vez utiliza os atributos de QoS no nível de serviços. Este trabalho oferece uma abordagem de monitoramento de coreografias integrada nas três camadas, no entanto, as restrições de QoS definidas no SLA são rígidas. Na pesquisa proposta neste documento, as restrições de QoS são definidas de maneira probabilística, e por conta disso o monitoramento também.

Linguagens de coreografia de serviços, tais como BPMN, Let's Dance e BPEL4Chor não aparecem em trabalhos relacionados com QoS na literatura. As buscas realizadas durante a escrita deste documento não retornaram nenhuma proposta de integrar um modelo de QoS no BPMN, que é um padrão para modelar processos de negócios colaborativos, incluindo coreografias. Existem várias propostas de modelos de QoS em serviços Web e orquestrações, mas poucas em coreografias, e ainda menos nas três camadas e de maneira integrada. Além disso, as poucas abordagens de monitoramento definem restrições de QoS de maneira rígida, o que não reflete a dinamicidade dos atributos de QoS dos serviços Web, e focam principalmente no monitoramento do tempo de resposta.

5.6.2 Monitoramento de Serviços Usando SLAs Probabilísticos

Em [RBJ09], é proposta uma técnica estatística para especificar contratos de QoS de maneira probabilística por meio de SLAs não rígidos (*soft SLA*). Os autores em [RBJ09] também propõem uma forma de realizar um monitoramento contínuo de orquestrações de serviços Web. Sua proposta pretende diminuir os alarmes falsos, e aumentar o número de detecções corretas. Tanto na especificação de SLAs probabilísticos quanto no monitoramento, são utilizados quantis das distribuições de probabilidade, e por conta disso, é utilizado o método de Monte-Carlo para obter parâmetros e valores para a configuração do monitoramento antes da sua execução. Assim como nas outras abordagens, o trabalho em [RBJ09] está focado no tempo de resposta, e dado que trata até orquestrações, atributos de rede como a latência e largura de banda não são tão importantes [TAM00]. Esta proposta de pesquisa, com relação a contratos probabilísticos, baseia-se fortemente no trabalho apresentado em [RBJ09], porque ele é dos poucos trabalhos que leva em consideração SLAs probabilísticos no monitoramento de serviços Web. Porém, na proposta desta pesquisa, a definição de SLAs e o monitoramento probabilístico serão realizados para coreografias de serviços Web. Outra diferença é que serão levados em conta outros atributos de QoS além do tempo de resposta. Esta pesquisa e o trabalho apresentado em [RBJ09] também são similares pela utilização de métodos de testes estatísticos para detectar a violação de SLAs.

[3]GSPN: Generalized Stochastic Petri Net

Em [ZYZ10], é proposta uma abordagem para modelar a função densidade de probabilidade (PDF[4]) da QoS de um serviço Web, baseada em um método estatístico não paramétrico. Os autores em [ZYZ10] fundamentam que a PDF é a melhor maneira de refletir as características dinâmicas dos atributos de QoS em serviços Web, e mostram por meio de simulações que sua abordagem é mais precisa do que outras, tais como distribuições Normais e distribuições *T Location-Scale*. Além disso, são fornecidas fórmulas para estimar a distribuição de QoS de uma composição de serviços a partir das PDFs dos serviços componentes. Diferentemente desta proposta de pesquisa, o trabalho em [ZYZ10] não cita a maneira de especificar restrições de QoS em SLAs, e nem como realizar monitoramento.

5.7 Considerações Finais

Neste Capítulo foram descritos os trabalhos que possuem alguma relação com o monitoramento de atributos de QoS em coreografias de serviços Web. Pode-se notar que há poucos trabalhos sobre modelos de QoS em coreografias, e ainda menos, aqueles que agem de maneira integrada com orquestrações e serviços Web. Existem poucos trabalhos relacionados com monitoramento de coreografias de serviços Web baseado em atributos de QoS. Também são poucas as abordagens que propõem um monitoramento baseado em contratos SLA de atributos de QoS de maneira probabilística.

A maioria dos trabalhos relacionados focam no atributo de tempo de resposta só, entretanto, no contexto de coreografias de serviços, tornam-se importantes os atributos de rede tais como o atraso e a largura de banda, já que uma coreografia envolve vários participantes que podem possuir arquiteturas e infraestruturas de rede heterogêneas e pertencentes a domínios diferentes, fazendo da rede um item fundamental para que a comunicação entre os serviços seja possível.

No entanto, no presente trabalho se desenvolveu um monitor centralizado de coreografias de maneira não intrusiva levando em consideração o modelo de multi-camada que inclui a QoS e SLA. Porém, já que não existem infraestruturas maduras o suficiente para habilitar a pesquisa em definição de restrições e monitoramento de coreografias de serviços Web, desenvolveu-se um simulador de coreografias com suporte de QoS. O próximo capítulo descreve a construção do simulador e a metodologia utilizada para cumprir com os objetivos desta pesquisa.

[4]PDF: *Probability Density Function*

Capítulo 6

Metodologia

Os capítulos anteriores apresentaram e descreveram os conceitos básicos acerca de serviços Web, computação orientada a serviços, composição de serviços, tópicos acerca de coreografias, qualidade de serviço, SLA e monitoramento em coreografias de serviços Web. A dissertação de mestrado neste documento tem como objetivo o monitoramento e a detecção de violações de SLAs baseadas em restrições probabilísticas de QoS em coreografias de serviços Web. Assim, as seções a seguir apresentam com mais detalhe a proposta. Primeiro uma visão geral na definição de restrições de QoS e o monitoramento de coreografias. Depois, apresenta-se a construção do simulador de coreografias para alavancar esta pesquisa. Finalmente, apresentam-se as técnicas para definir os requerimentos de QoS e monitoramento para realizar a detecção de violações de SLA em coreografias.

6.1 Visão Geral

A Figura 6.1 ilustra a arquitetura para a definição de SLAs e monitoramento de coreografias de serviços. Apresenta-se uma coreografia de três participantes (A, B e C) que interagem entre si. Cada um dos participantes possui uma implementação local por meio de orquestrações de serviços, que por sua vez "realizam" a coreografia. Um SLA é definido por cada dupla de participantes que interagem. Por exemplo, na interação dos participantes A e B foi definido um SLA (SLA 1) composto pelas restrições de QoS (rótulo 1 da Figura 6.1) do tempo de resposta no fornecimento das operações que acontecem nessas interações. Tais restrições são definidas de maneira probabilística utilizando uma distribuição de probabilidade. Desta maneira, podem-se definir SLAs probabilísticos para cada dupla de participantes que interagem (SLA 2 e SLA 3) de acordo com a especificação da coreografia. Cabe ressaltar que as restrições de QoS definidas no SLA de um serviço podem ser o resultado da agregação de QoS de otros serviços oferecidos por outros participantes de acordo com as suas dependências.

O monitoramento da coreografia (rótulo 2 da Figura 6.1) e a detecção de violações de SLAs são realizados uma vez que os SLAs probabilísticos tenham sido definidos. Em tempo de execução, o monitor deve realizar as medições dos atributos de QoS dos serviços Web, estimar suas distribuições de probabilidade, realizar a agregação de QoS mediante as distribuições de probabilidade e comparar a distribuição agregada com a distribuição de probabilidade especificada no SLA. Dessa maneira, o monitor realiza as verificações necessárias para detectar violações de SLA.

6.2 Preliminares

O modelo de interação para especificar coreografias foi adoptado nesta pesquisa. Escolheu-se essa abordagem já que esse modelo representa melhor as interações entre participantes e porque possui menos complexidade do que no modelo de interconexão. Assim, o padrão BPMN2 é levado

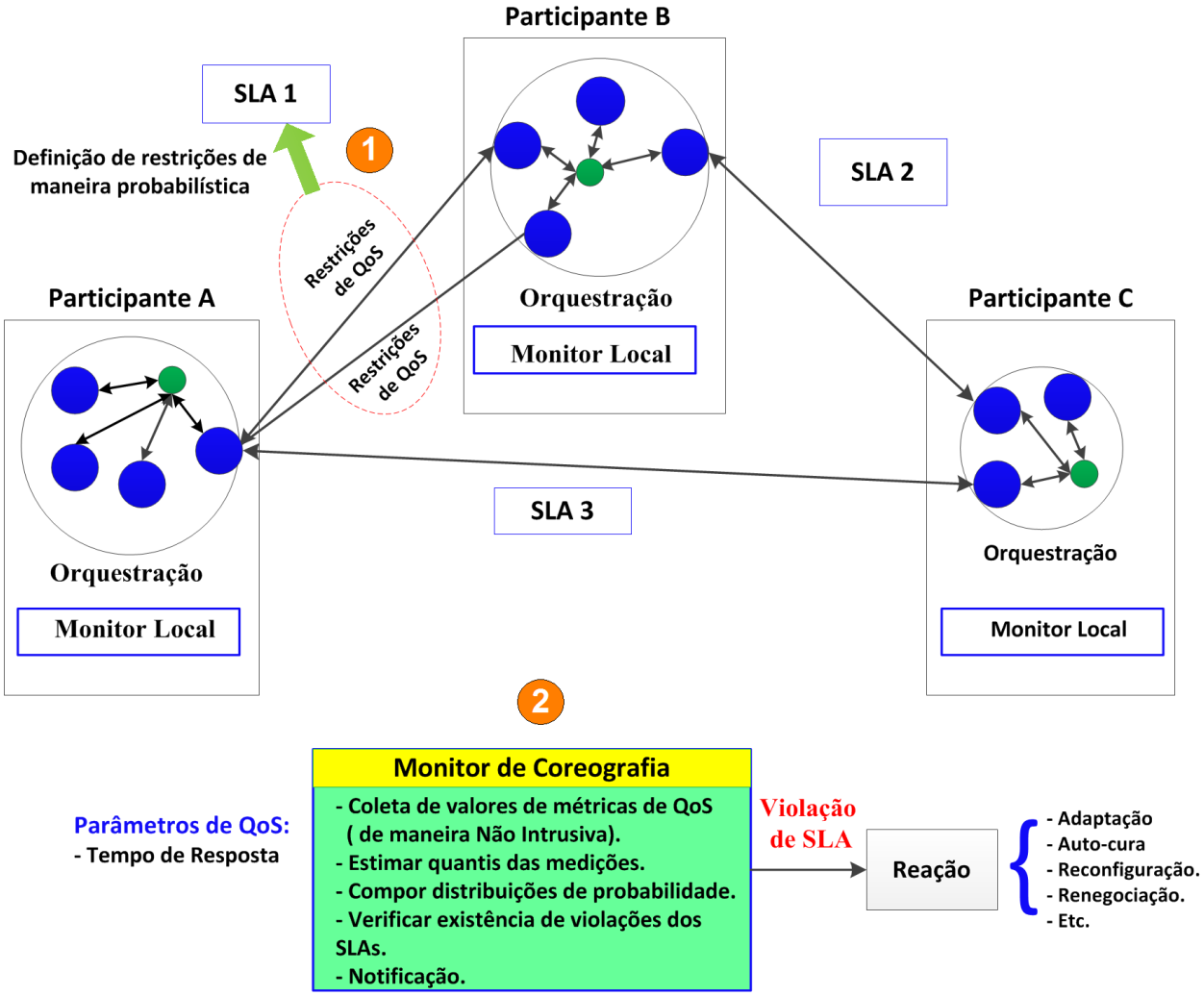

Figura 6.1: *Arquitetura do monitoramento de coreografias baseado em SLAs probabilísticos*

em consideração por suportar o modelo interação. A Figura 6.2 mostra os elementos BPMN considerados no desenvolvimento desta pesquisa. Considerou-se somente um subconjunto do total de elementos, já que são suficientes para avaliar as técnicas e abordagens propostas.

A Figura 6.3 mostra as três principais etapas envolvidas para realizar o monitoramento de coreografias. Primeiro, precisa-se definir os requerimentos de QoS por meio de métodos análiticos ou simulações. Depois, especifica-se o contrato de QoS (SLA) por meio de alguma distribuição de probabilidade. Por fim, configuram-se os parâmetros de monitoramento, se realiza a coleta e agregação de QoS, e se realizam as verificações para detectar as violações de SLAs. Dessa maneira, para atingir o objetivo de realizar a detecção de violações de SLAs em coreografias de serviços Web, desenvolveu-se as seguintes atividades nesta pesquisa:

- Definição de requerimentos de QoS de maneira análitica, por meio da avaliação de desempenho de coreografias usando Redes de Petri Estocásticas Generalizadas (GSPN).
- Construção de um simulador para rodar coreografias de serviços. O simulador de coreografias possui suporte de QoS, e acima dele um monitor centralizado foi desenvolvido.
- Definição de requerimentos de QoS de maneira probabilística usando o simulador de coreografias.
- Especificação de contratos probabilística baseado nos requerimentos definidos.
- Monitoramento de SLAs probabilísticos e a detecção de violações de SLA.

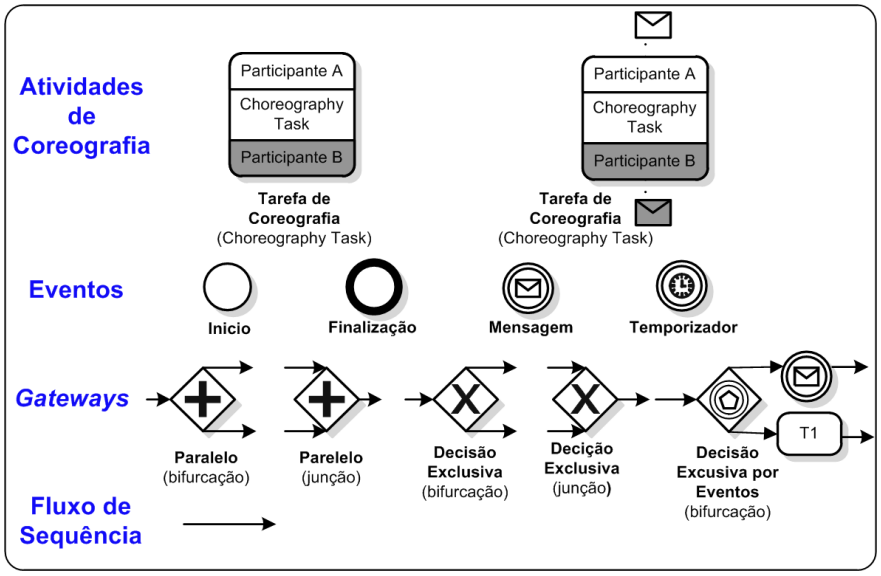

Figura 6.2: *Elementos BPMN para modelagem de coreografias que são levados em consideração.*

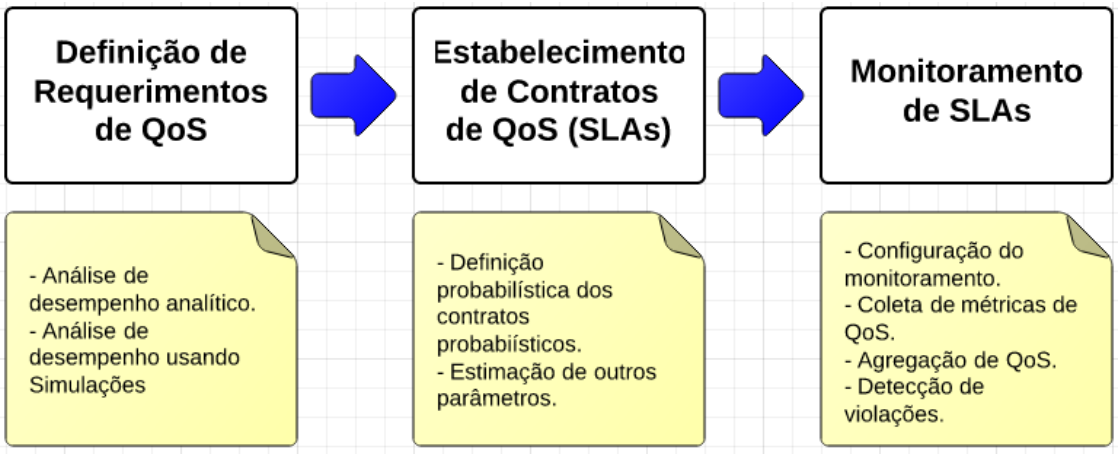

Figura 6.3: *Interação de serviços a partir de interações atômicas do BPMN2.*

6.2.1 Modelo de QoS

Os atributos de QoS considerados neste trabalho estão envolvidos em interações de serviços, isto é, de acordo a um modelo de requisição e resposta para um serviço (individual ou composto). Essas interações entre serviços estão baseadas nas interações atômicas (atividades de coreografia) do BPMN2. A Figura 6.4 mostra o mapeamento das atividades de coreografia para seu equivalente em interações de serviço, de maneira a definir aí o modelo de QoS (atributos, métricas, cálculos, entre outros).

A Figura 6.5 mostra os atributos de QoS básicos envolvidos em uma invocação de um serviço tais como, o tempo de comunicação na requisição, o tempo de execução, o tempo de comunicação na resposta e o tempo de resposta. O tempo de comunicação depende de atributos de QoS de rede como a largura de banda e a latência de rede; o tempo de resposta depende do tempo de execução e do tempo de comunicação (de requisição e de resposta). Além do mais, no caso de serviços compostos o cálculo dos atributos de QoS depende das medidas dos outros atributos de QoS e das dependências com outros serviços.

Os atributos de QoS podem ser calculados da seguinte forma:

- **Tempo de comunicação**: $t_c = L_{ij} + S/Bij$. Onde S é o tamanho da mensagem, e L_{ij} e Bij são a latência e largura de banda de rede entre o ponto i e o ponto j.

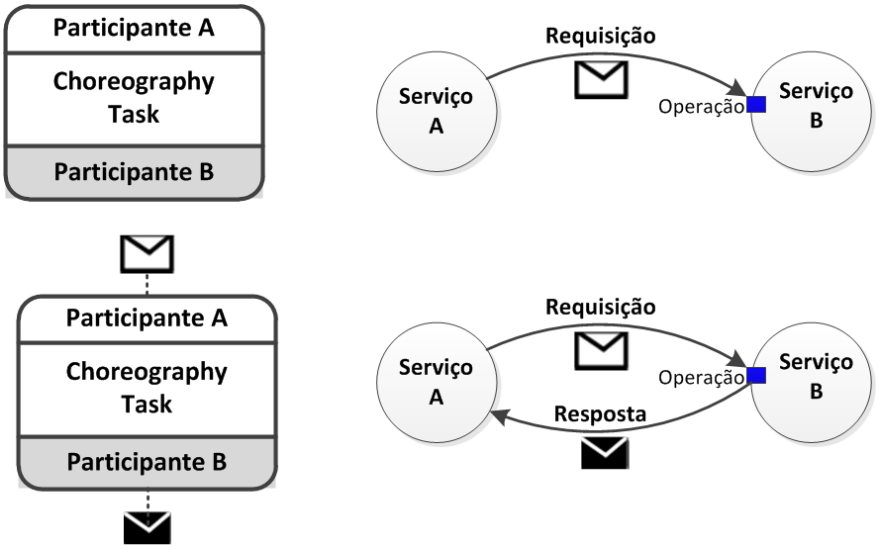

Figura 6.4: *Interação de serviços a partir de interações atômicas do BPMN2.*

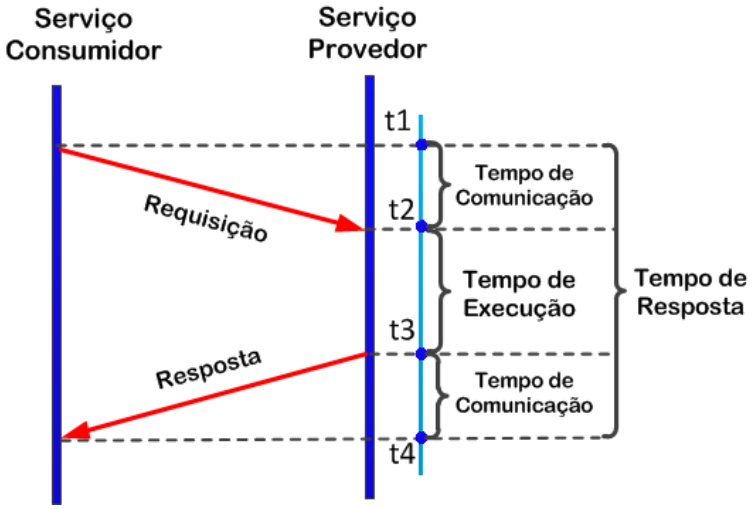

Figura 6.5: *Atributos de QoS em uma interação com um serviço Web.*

- **Tempo de execução**: $t_{\text{execução}} = t_3 - t_2$.
- **Tempo de resposta**: $t_R = t_{\text{execução}} + t_{c1} + t_{c2}$. Onde o t_{c1} e t_{c2} são os tempos de comunicação na requisição e na resposta respectivamente.
- **Tempo de execução efetivo**: $t_{execuo} = t_{Rcomposto} + t_{execuo\_individual}$. Onde $t_{Rcomposto}$ é o tempo de resposta acumulado dos serviços que são dependências do serviço atual, e $t_{\text{Execução}\_individual}$ é o tempo de execução do serviço atual.
- **Tempo de resposta composto**: $t_{Rcomposto} = F(aggregationType, dado, t_{R1}, ..., t_{Rk})$. Onde os $t_{R1}, ..., t_{Rk}$ são os tempos de resposta dos serviços que são dependências do serviço atual, F é a função de agregação que depende do tipo de padrão de *workflow* $aggregationType$, e *data* é a informação associada. Os *gateways* atualmente suportados são a sequência, paralelismo e a decisão exclusiva (ver Figura 6.2):
  - **Sequência**: $F(SEQUENCE, dado, t_{R1}) = t_{R1}$. Onde t_{R1} é o serviço em sequência do serviço atual.
  - **Paralelismo**: $F(PARALLEL, dado, t_{R1}, ..., t_{Rk}) = max\{t_{R1}, ..., t_{Rk}\}$. Onde $t_{R1}, ..., t_{Rk}$ são os tempos de resposta dos serviços que são dependências do serviço atual.

– **Decisão exclusiva**: $F(EXCLUSIVE, dado, t_{R1}, ..., t_{Rk}) = t_{Ry}$. Onde o t_{Ry} é o tempo de resposta do serviço escolhido de acordo com o valor de *data*.

6.2.2 Modelo de Falhas

Um modelo de falhas descreve os tipos de falhas que podem ocorrer em um sistema enquanto está sendo executado, e ajuda a determinar quais mecanismos de tolerância a falhas deveriam ser aplicados [LLM+10]. Levando em consideração o trabalho em [LLM+10], as falhas para serviços Web podem-se classificar em quatro tipos: lógicas, de sistema, de conteúdo e de nível de serviço (SLA). A seguir, uma breve descrição desses tipos:

- **Falhas Lógicas**: Detectadas na lógica definida na especificação da composição.
- **Falhas de Sistema**: Surgem no suporte do ambiente de execução e estão relacionadas a falhas nos computadores, rede, sistema operacional, etc.
- **Falhas de Conteúdo**: Devido a dados ou mensagens corrompidos.
- **Falhas de SLA**: Devido a violações de QoS especificados em um SLA.

No contexto de coreografias, já que há serviços compostos, os atributos de QoS são individuais e agregados de acordo com os padrões de fluxo de trabalho e suas dependências. Um monitor deve ficar responsável pela coleta, medição e agregação de tais atributos, assim como por acompanhar a execução da coreografia e detectar violações de restrições de QoS.

6.3 Definição de requisitos de QoS analiticamente

Antes de desenvolver a pesquisa sobre a infraestrutura para rodar coreografias (simulador), desenvolveu-se uma maneira de definir requisitos de QoS de maneira analítica usando GSPNs (Rede de Petri Estocástica Generalizada). Assim, uma nova metodologia para avaliar o desempenho de coreografias para definir requisitos de QoS foi proposta. A avaliação é realizada na fase de modelagem da coreografia usando GSPNs. A proposta baseia-se na execução dos seguintes passos:

1. Mapeamento de uma coreografia especificada em BPMN 2.0 para uma GSPN. A GSPN inclui a representação intermediária da coreografia e o modelo de QoS que leva em consideração as restrições de tempo e comunicação das interações entre os participantes (cada participante representa um *peer* que está ativo durante o *enactment* da coreografia).
2. Definição de pesos nas transições com restrições de tempo na GSPN. Nesta etapa, as distribuições de probabilidade e pesos nas interações relacionadas com a comunicação e troca de mensagens são definidas.
3. Simulações de cenários, um cenário com uma configuração mínima de falhas e outro cenário com uma maior possibilidade de falhas. As simulações são utilizadas já que as coreografias no mundo real representam processos complexos, que geram complexas redes de Petri com muitos estados que tornam difícil realizar avaliações analíticas [OLAR09].

6.3.1 Definição formal de coreografias specificadas em BPMN

Com base na Figura 6.2 e no trabalho apresentado em [DDO07], as coreografias de processos especificadas em BPMN2 podem ser definidas como:

Definição 1. *Uma coreografia de processos em BPMN2 é a tupla:*
$PC = (\mathcal{O}, \mathcal{A}, \mathcal{E}, \mathcal{G}, \mathcal{T}, \{e^S\}, \mathcal{E}^I, \{e^E\}, \mathcal{E}^{I_M}, \mathcal{E}^{I_T}, \mathcal{G}^F, \mathcal{G}^J, \mathcal{G}^X, \mathcal{G}^M, \mathcal{G}^V, \mathcal{F})$ *onde:*

- *$\mathcal{O}$ é conjunto de "objetos", que é particionado em três conjuntos disjuntos: atividades ($\mathcal{A}$), eventos $\mathcal{E}$ e* gateways *$\mathcal{G}$.*
- *$\mathcal{A}$ é conjunto de atividades de coreografia (choreography tasks) $\mathcal{T}$, sub-coreografias e call Choreographies. No entanto, sub-coreografias e call choreographies não são levados em consideração nesta pesquisa. Por isso, $\mathcal{A}$ é igual que o conjunto $\mathcal{T}$.*
- *$\mathcal{E}$ é conjunto de eventos. Os eventos são particionados em três conjuntos disjuntos: eventos de Inicio $e^{\mathcal{S}}$, eventos Intermediários $\mathcal{E}^{\mathcal{I}}$ e eventos de Finalização $e^{\mathcal{E}}$.*
- *$\mathcal{E}^{\mathcal{I}}$ é particionado nos conjuntos disjuntos de eventos de menssagens $\mathcal{E}^{I_M}$ e eventos de temporizador (timer) $\mathcal{E}^{I_T}$.*
- *$\mathcal{G}$ é o conjunto de gateways e é particionado em cinco conjuntos disjuntos de gateways: "parallel forks" $\mathcal{G}^F$, "parallel join gateways" $\mathcal{G}^J$, "data-based XOR gateways" $\mathcal{G}^X$, "XOR merge gateways" $\mathcal{G}^V$ e "event-based XOR gateways" $\mathcal{G}^M$.*
- *$\mathcal{F} \subseteq \mathcal{O}x\mathcal{O}$ é a relação de fluxo de controle, isto é, um conjunto de fluxos de sequência que conectam objetos.*

Para analisar o comportamento não funcional em coreografias é necessário definir os atributos de QoS a considerar. Para tanto, consideraram-se atributos de QoS para três entidades: serviço, rede e mensagens. No serviço, o atributo de qoS considerado é o tempo necessário para completar uma requisição (tempo de resposta). Na rede, consideram-se os atributos como o atraso de rede e os erros de comunicação, já que influem significantemente no tempo de resposta em uma coreografia. Nas mensagens, considera-se o formato da mensagem, já que necesitam ser validados.

6.3.2 Mapeamento de coreografias em BPMN 2.0 para GSPNs

A Figura 6.6 apresenta o mapeamento de *gateways* e eventos para seus respectivos módulos de rede de Petri. A Figura 6.7 apresenta o mapeamento dos dois tipos de tarefas de coreografia (blocos de construção de coreografias já que geram mensagens e invocações de serviços). Nessa Figura, as posições A, A_2 e B representam os participantes. As posições C_{init}, C_{end}, C_{init2}, C_{end2}, C_{endOk} e C_{endOk2} representam o começo, o final e a correta validação da comunicação entre dois participantes. As transições T_{send} e $T_{receive}$ representam as ações de envio e recebimento de mensagens. As transições T_{msg} representam a transmisão de mensagens por um canal de comunicação.

O passo 1 da metodologia é realizada pelo Algoritmo 1. O algoritmo mapeia uma coreografia de processos especificados em BPMN 2.0 para uma GSPN incluindo o modelo de QoS. Cada elemento BPMN é mapeado para seu respectivo modulo de rede de Petri (módulo GSPN) e esses módulos são compostos por meio de uma função de acordo com os fluxos de sequência ou *gateways* que os conectam. Finalmente, os eventos de Inicio e de Finalização são adicionados na rede Petri atualmente construída.

6.4 Simulador de Coreografias de Serviços

Atualmente, implementar e executar uma coreografia de serviços real é uma tarefa complexa já que as tecnologias para suportar esse paradigma de composição de serviços estão ainda imaturas, especialmente pela falta de motores de execução cientes de coreografia [KELL10]. Assim, os mecanismos para medir parâmetros de QoS, e estabelecer requisitos de QoS não estão bem desenvolvidos para coreografias.

6.4.1 Trabalhos Relacionados

Existem poucas infraestruturas para implementação de coreografias de serviços Web [BBRW09]. O Pi4SOA [ZTW+06] é um arcabouço que fornece um editor para modelagem de coreografias

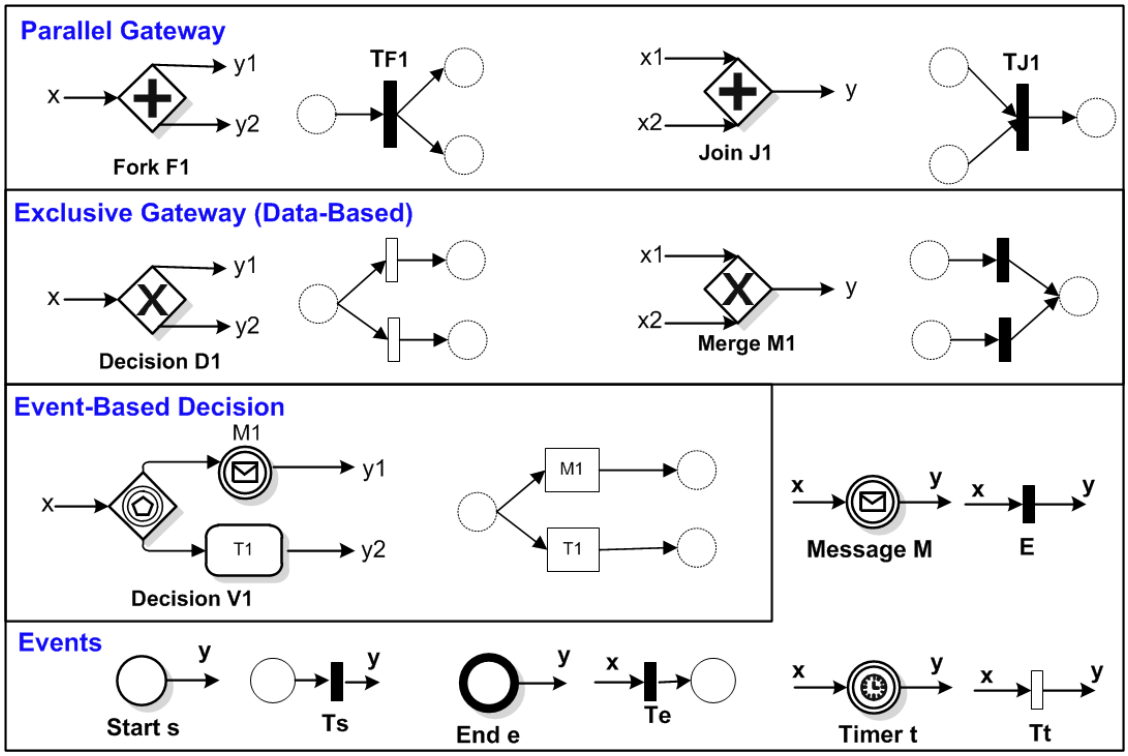

Figura 6.6: *Mapeamento de eventos e* gateways *para módulos de redes de Petri*

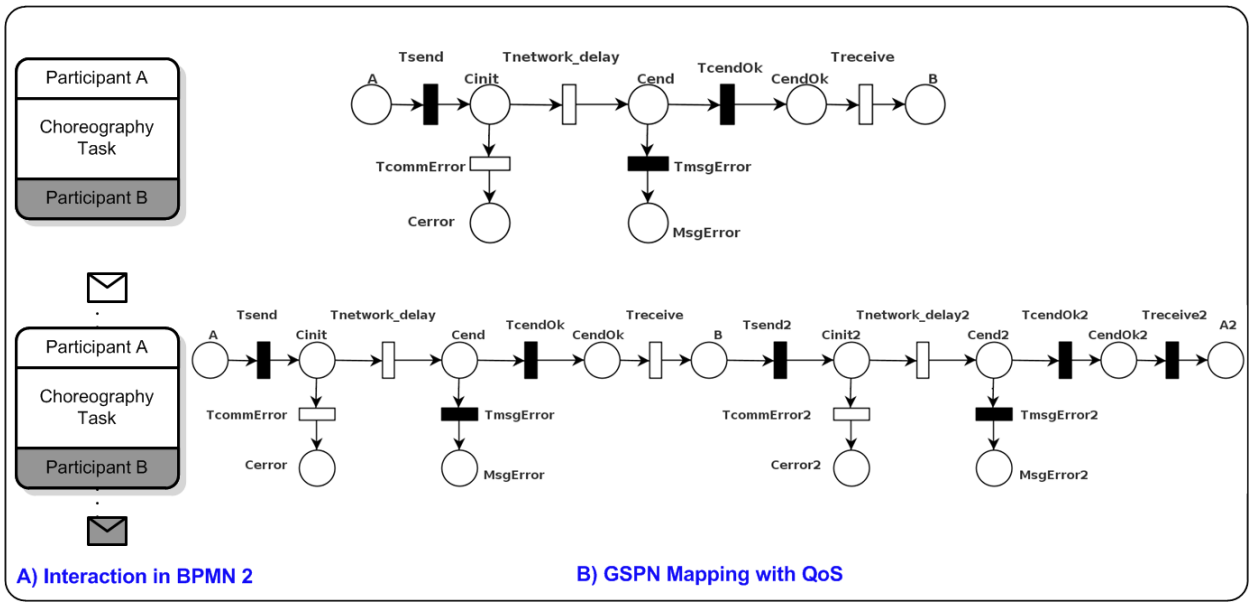

Figura 6.7: *Mapeamento de dois tipos de tarefas de coreografia para módulos de redes de Petri com suporte de QoS*

em WS-CDL[1], e realiza verificações, validações e simulações. Porém, não suporta execução de coreografias e as suas simulações servem apenas para encontrar inconsistências na especificação. O WS-CDL+ [KWH07] é uma proposta de um motor de execução de coreografias especificadas em WS-CDL, mas foi implementado na forma de protótipo e somente a versão 1.0 foi lançada. O OpenKnowledge [BBRW09] é um arcabouço que fornece a capacidade de rodar sistemas distribuídos em uma arquitetura P2P (ponto a ponto), podendo rodar coreografias também, mas de maneira limitada.

[1]WSCDL: Linguagem de especificação de coreografias proposto pela W3C

Algorithm 1 Mapeamento de uma coreografia especificada em BPMN 2.0 para uma GSPN com suporte de QoS

Entrada: Coreografia de Processos $PC = (\mathcal{O}, \mathcal{A}, \mathcal{E}, \mathcal{G}, \mathcal{T}, \{e^S\}, \mathcal{E}^I, \{e^E\}, \mathcal{E}^{I_M}, \mathcal{E}^{I_T}, \mathcal{G}^F, \mathcal{G}^J, \mathcal{G}^X, \mathcal{G}^M, \mathcal{G}^V, \mathcal{F})$ em BPMN 2.0.
Saída: $GSPN_{QoS}$, uma rede de Petri estocástica generalizada com suporte de QoS.
Considerar $CT_i \in \mathcal{T}$, $G_j \in \mathcal{G}$ e $E_k \in \mathcal{E}$. $i, j, k \in \mathbb{N}$
Seja $PNQoS(CT_i)$ uma função que de acordo com o tipo de CT_i e de acordo com regras de mapeamento como ilustrado na Figura 6.7 retorna uma GSPN incluindo QoS.
Seja $PNQoS(G_j)$ uma função que de acordo com o tipo de G_j e de acordo com regras de mapeamento como ilustrado na Figura 6.6 retorna uma GSPN com suporte de QoS.
Seja $PNQoS(E_k)$ uma função que de acordo com o tipo de E_k e de acordo com regras de mapeamento como ilustrado na Figura 6.6 retorna uma GSPN com suporte de QoS.
Seja $\oplus$ o operador binario de composição de dois módulos de GSPNs e que retorna outra GSPN.
$GSPN_{QoS} \leftarrow Empty\ Petri\ Net$
for $CT_i \in \mathcal{T}$ **do**
    $GSPN_{QoS} \leftarrow GSPN_{QoS} \oplus PNQoS(CT_i)$
    Adicionar uma transição de tempo de chegada antes de $GSPN_{QoS}$.
end for
for $G_j \in \mathcal{G}$ **do**
    $GSPN_{QoS} \leftarrow GSPN_{QoS} \oplus PN(G_j)$
end for
for $E_k \in \mathcal{E}$ **do**
    $GSPN_{QoS} \leftarrow GSPN_{QoS} \oplus PN(E_k)$
end for
Adicionar uma posição de Inicio e uma transição immediata no começo de $GSPN_{QoS}$.
Adicionar uma posição de finalização e uma transição immediata no final de $GSPN_{QoS}$.
return $GSPN_{QoS}$

Muitos simuladores para sistemas e ambientes distribuídos foram propostos. Por exemplo, o arcabouço GridSim [BMCC02], o Pi4SOA [ZTW+06] e o arcabouço SimGrid [CLQ08]. O arcabouço GridSim [BMCC02] é um motor de simulação de ambientes distribuídos baseado em eventos. Ele implementa entidades para emular usuários. As requisições dos usuários são escalonadas por meio de um *broker* que os aloca nos recursos de simulação. O SimGrid [CLQ08] é um arcabouço para simular diversos sistemas distribuídos e permite avaliar mecanismos de *clusters* e grades.

Como podemos notar, não há soluções baseadas em simulações para apoiar a execução de coreografias e menos ainda com suporte a QoS. Por conta disso, em [GKB12] se desenvolveu um protótipo de um simulador para demonstrar que coreografias são mais eficientes do que orquestrações de serviços. Contudo, esse simulador não suporta experimentos de coreografias com o objetivo de avaliar mecanismos relacionados com QoS ou SLA.

Nosso trabalho diferencia-se dos demais por apresentar um novo simulador para coreografias de serviços Web que permite a avaliação de mecanismos de garantia de QoS e de SLA. Além disso, também é apresentada uma metodologia para estabelecer requisitos de QoS para coreografias de serviço.

6.4.2 Desenvolvimento

A Figura 6.8 mostra as requisições enviadas e as respostas de um serviço composto. Essas interações (requisições e respostas) são registradas mediante eventos por parte de um serviço ou cliente solicitador (eventos 1 e 4) e eventos por parte das dependências (eventos 2 e 3). Os eventos indicam quando e quais atributos de QoS devem ser medidos e garantem a ordem para garantir valores consistentes. Os atributos de QoS com as suas métricas e tipos de falhas associadas são apresentados na Tabela 6.1.

A simulação é utilizada pois a implementação e a execução de coreografias de serviços Web reais ainda é difícil por conta da imaturidade das tecnologias [KELL10]. Porém, implementar um

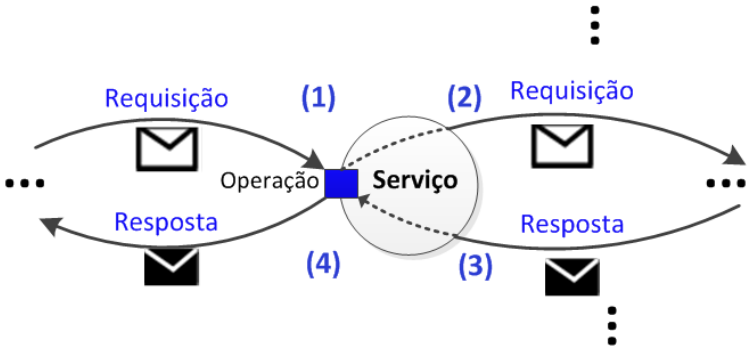

Figura 6.8: *Atributos de QoS calculados em um evento dado. (1) Recebendo requisições de um cliente ou serviço. (2) enviando requisições para um outro serviço. (3) recebendo resposta de um outro serviço (dependência). (4) enviando resposta para um cliente ou serviço solicitador.*

Tabela 6.1: *Modelo de QoS e de falhas*

Tipo	Atributo de QoS	Métrica	Tipo de Falha
Serviço	Tempo de Resposta	ms	Temporização, violação de QoS
Serviço	Vazão	#requisições/s	Serviço não disponível,violação de QoS
Mensagem	Formato da Mensagem	-	Probabilidade de falha
Comunicação	Latência	ms	Erro de comunicação/violação de QoS
Comunicação	Largura de Banda(máxima)	Mb/s	Erro de comunicação

simulador por completo é uma tarefa complexa. Por conta disso, decidiu-se usar um arcabouço de simulação existente, o SimGrid [CLQ08]. Como o SimGrid permite a simulação de ambientes distribuídos, ele é suficiente para servir como base para o nosso simulador de coreografias com suporte a avaliação de desempenho de atributos de QoS.

6.4.3 Arquitetura

A Figura 6.9 mostra a arquitetura do simulador de coreografias (ChorSim[2]) com suporte a QoS, onde cada bloco representa um componente do simulador. Um componente depende de um outro componente ou vários componentes que estão embaixo dele. Cada um dos componentes serão explicados a seguir. A base do ChorSim está construído sobre o arcabouço SimGrid, para suportar a definição de *hosts*, topologia de rede, comunicação entre serviços e a especificação do consumo de recursos. O "Motor de Execução de Coreografias" do ChorSim permite a criação de instâncias de coreografias e iniciar as interações dos diversos serviços envolvidos que resultarão em um grafo de requisições e um outro grafo de informações de QoS.

A topologia da coreografia (host, canais de comunicação e links) é configurada por meio de um arquivo XML de especificação de plataforma. Os serviços foram modelados como um conjunto de *threads* que recebem uma tarefa enviada através da rede, a executam e em seguida enviam outra tarefa através da rede para atuar como uma resposta da requisição.

O "Modelo de Coreografias" é construído a partir de uma especificação baseada em XML que define os serviços participantes, suas dependências e suas interações. Os recursos computacionais necessários para executar, a quantidade de *threads*, o tamanho médio das respostas dadas nas interações dos serviços e suas operações são configurados por meio de um arquivo XML de implantação, que está especificado de acordo como o modelo de coreografias. O arquivo de implantação serve também como base para construir o "Registro de Serviços".

O "Gerenciador de Requisições e QoS" gerencia o grafo de requisições e o grafo de informações de QoS de acordo com o avanço e execução na interação dos serviços. Acima desta infraestrutura o "Monitor de Coreografias" é desenvolvido, usando o "Modelo de Coreografias", o "Registro de Serviços" e o "Gerenciador de Requisições e QoS". Esse monitor é responsável pela medição dos atributos

[2] O repositório do ChorSim está disponível em https://github.com/choreos/ChoreographySimulator

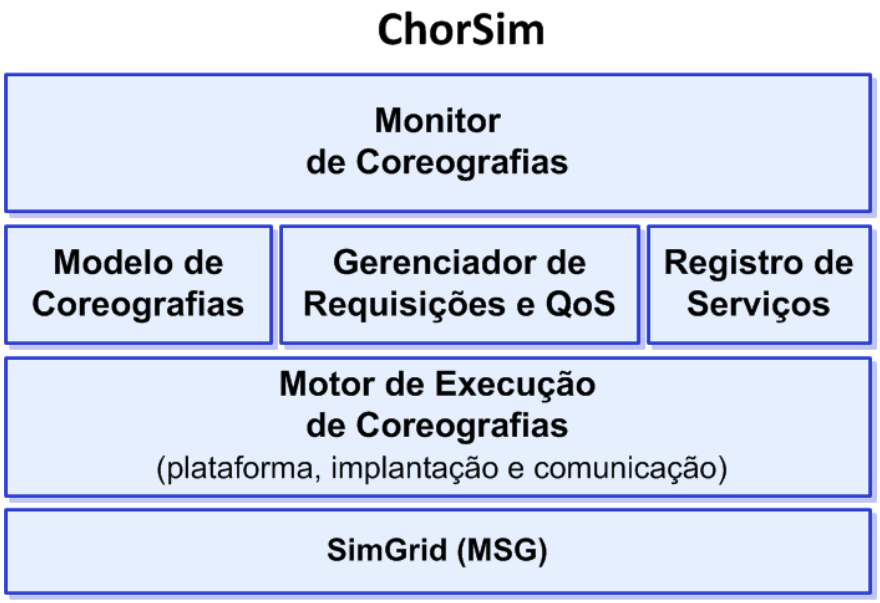

Figura 6.9: *Arquitetura do simulador de coreografias.*

de QoS dos serviços individuais e por agregá-los para calcular os atributos de QoS compostos. A agregação de QoS é realizada de acordo com as dependências, interações e os padrões de fluxo de trabalho (*gateways*) do modelo de coreografia. Por meio do monitor criam-se e gerenciam-se as instâncias de coreografias e seus respectivos identificadores para serem usados nas mensagens durante as interações.

6.4.4 Definição de requerimentos de QoS

Injeção de falhas é amplamente utilizado para validar a mecanismo de tolerância a falhas em sistemas computacionais. Em um ambiente de computação distribuída, os esforços de pesquisa têm desenvolvido maneiras sistemáticas de identificar falhas em tempo de design e assim projetar mecanismos de gestão de falhas. Simulações baseadas em dados reais são muitas vezes inadequadas, já que os dados podem não estar disponíveis e sua coleta pode exigir longos períodos de tempo de observação. Em vez disso, injeção de falhas é uma maneira prática e eficiente de validação, visto que consiste na inserção de falhas no sistema tanto no software e nos níveis de hardware [FPR09]. Com base nisso, utilizou-se injeção de falhas no simulador de coreografias (*ChorSim*) para obter maior diversidade nas medidas dos atributos de QoS (tempos de resposta, largura de banda) dos serviços individuais e do serviço composto.

Para definir quais requerimentos de QoS são necessários para definir um contrato, precisa-se realizar diversas simulações con diversas configurações para procurar gargalhos, cargas de trabalho , degradações, entre outros, acerca de determinados atributos de QoS. Além disso, para obter comportamentos mais realistas pode se usar distribuições de probabilidade. Porém, o SimGrid[3] não suporta a geração de eventos probabilísticos, por isso também não suporta a especificação de recursos computacionais nem de rede por meio de distruibuições de probabilidade.

Mesmo assim, no ChorSim conseguiu-se adicionar a funcionalidade de usar distribuições de probabilidade para serem utilizados para definir somente comportamentos dinâmicos no tempo de execução ou de processamento de serviços. Outros atributos como a largura de banda e latência foram excluídos para serem definidos como distribuições de probabilidade, porque os modelos que os suportam estão bem atrelados com a implementação do SimGrid. Contudo, para definir um comportamento variável da largura de banda suportado pelo SimGrid usou-se a periodicidade. Assim, diversos valores de largura de banda são definidos em intervalos de tempo que estão definidos

[3] A última versão do Simgrid é a 3.8

em um período de tiempo. Dessa maneira, no próximo Capítulo se mostrarão os experimentos realizados com tempos de execução definidos como distribuições de probabilidade e largura de banda definidos com valores váriaveis dentro de um período.

6.5 Especificação de SLAs Probabilísticos

Nesta pesquisa, para cada relacionamento existente entre os participantes de uma coreografia serão especificadas restrições de QoS de maneira probabilística. Tais restrições de QoS tornarão-se um contrato (SLA) probabilístico.

Assim, um contrato é especificado por meio de uma função de distribuição de probabilidade

$$F(x) = P_r(\delta \leq x)$$

Onde P_r é a probabilidade correspondente a uma restrição sobre algúm parámetro de QoS. Assim, δ é um parâmetro de QoS (por exemplo, a largura de banda efetiva), e $x \geq 0$. Então, o contrato ou restrição será estabelecido como um conjunto finito de *quantis* dos parâmetros de QoS. A forma de definir essas restrições utilizando quantis será baseada no trabalho apresentado em [RBHJ08], o qual propõe uma técnica para especificar SLAs probabilísticos entre serviços utilizados em uma orquestração.

Esta pesquisa realizará a agregação das distribuição de probabilidade dos parâmetros de QoS de acordo com os padrões de interação de coreografia, de modo que a restrição ou contrato fique estabelecida no SLA de uma dupla de participantes da coreografia. Para estimar os quantis a utilizar nas restrições de QoS nos SLAs dos participantes, precisa-se compor as distribuições de probabilidade das restrições de QoS dos serviços individuais que são dependências do serviço composto.

Pretende-se modificar o procedimento do método de Monte-Carlo para orquestrações proposto em [RBJ09], para suportar também os padrões de interação das coreografias além de orquestrações. O conjunto de restrições de QoS por meio de quantis, comporão um SLA, os quais serão descritos utilizando o padrão WSLA [KL03], introduzindo nele um conjunto de quantis por cada restrição de QoS que for acordada.

6.6 Monitoramento e Detecção de Violações de SLA

O monitor proposto nesta pesquisa deverá continuamente detectar possíveis violações de SLA. O monitor deve atingir o menor número de alarmes falsos e detectar o maior número de violações de SLA certas. O monitoramento usa métodos estatísticos para verificar se o desempenho observado se desvia do desempenho acordado no SLA.

O objetivo será medir os valores das métricas de QoS de um serviço S de um participante. Depois os valores serão comparados com a restrição de QoS F_s definida no SLA não rígido. F_s é a distribuição mediante quantis do parâmetro δ de S. Seja Δ um conjunto finito de amostras dos valores medidos de algum parâmetro de S. As seguintes equações de [RBJ09] são utilizadas para ilustrar o que se pretende fazer:

$$F'_{s,\Delta}(x) = \frac{|\{\delta, \delta \in \Delta \leq x\}|}{|\Delta|} \tag{6.1}$$

Onde $F'_{s,\Delta}(x)$ é a função distribuição de probabilidade empírica, que define a proporção das amostras dos valores medidos coletado pelo monitor do parâmetro δ que são menores que x dentro do conjunto Δ. Daí, informalmente a restrição é cumprida se:

$$\forall x \in R^+ : F'_{s,\Delta}(x) \geq F_s(x) \tag{6.2}$$

Onde R^+ é o conjunto de números reais positivos. De forma equivalente, a violação de uma

restrição de QoS acontece se:

$$\exists x \in R^{+} : F'_{s,\Delta}(x) < F_s(x) \tag{6.3}$$

Dado que em (2) e (3), $F'_{s,\Delta}(x)$ pode variar aleatoriamente ao redor de $F_s(x)$, especialmente quando Δ é bem pequeno, precisa-se de uma zona de tolerância para tais desvios. Desta maneira a condição de violação pode ser formulada como:

$$sup_{x \in R^{+}}(F'_{s,\Delta}(x) - F_s(x)) \geq \lambda \tag{6.4}$$

Onde λ é um parâmetro positivo que define a zona de tolerância. Um valor pequeno de λ melhora a precisão do monitoramento na detecção de violações, mas acrescenta o risco de alarmes falsos.

Nesses pontos descritos acima, precisa-se calibrar alguns parâmetros tais como o tamanho de Δ e o valor adequado de λ. Antes de utilizar essas fórmulas, precisa-se também de um método para agregar os quantis dos valores medidos dos parâmetros de QoS dos serviços individuais até achar os valores acumulados da coreografia. Para tanto, esta pesquisa levará em consideração padrões de interação de coreografias [BDH05].

6.6.1 Detecção de violações de SLA

Os testes estatísticos de dominância estocástica estabelecem o quadro matemático adequado para declarar corretamente e resolver o problema de monitoramento. Aquí, o problema é estabelecido a seguir:

Seja F_S a função de distribuição cumulativa prometida pelo contrato de um serviço S, e seja F'_S a função de distribuição cumulativa atual do serviço S. O problema é decidir entre duas hipóteses:

$$\begin{aligned} H_0 : &\quad \forall x, F_S(x) >= F'_S(x) \\ contra : & \\ H_1 : &\quad \exists x, F_S(x) < F'_S(x) \end{aligned} \tag{6.5}$$

Onde na hipótese nula (H_0) o contrato é cumprido e na hipótese alternativa (H_1) o contrato é violado. Dado que o comportamento dos atributos de QoS dos serviços individuais pode ter diversas distribuições de probabilidade, o comportamento de um serviço composto pode resultar também bem diferente. Assim, O teste de Kolmogorov-Smirnov de apenas um lado de dois amostras[4] é utilizado para aceitar ou rejeitar H_0:

$$[d, p] = kstest(X_{contract}, X_{monitoring}, KS_{side}) \tag{6.6}$$

Onde *kstest* é a função que desempenha o teste de Kolmogorov-Smirnov (KS)de um lado de dois amostras, que compara as distribuições dos valores nos vetores $X_{contract}$ e $X_{monitoring}$. O vetor $X_{contract}$ corresponde ao contrato e o vetor $X_{monitoring}$ é o conjunto de amostras dos valores de um atributo de QoS (por exemplo, tempo de reposta) obtidos por meio da composição de valores dos serviços individuais usando o simulador de coreografias. A hipótese nula consiste em que $X_{contract}$ e $X_{monitoring}$ são da mesma distribuição continua e a hipótese alternativa consite em que $X_{contract}$ e $X_{monitoring}$ são de diferentes distribuições. O argumento KS_{side} é um parametro cujo valor pode ser *greater*, *less* ou *two-sided*. Para a detecção de violações usamos os valores *greater* e *less*. Por outro lado, d é a distancia entre as duas distribuições cumulativas de $X_{contract}$ e $X_{monitoring}$ e p é o "*p-value*" do teste estatístico que tem valores entre $[0, 1]$.

Então, para a detecção de vioalções usam-se:

$$\begin{aligned} [D^{+}, p^{+}] &= kstest(X_{contract}, X_{monitoring}, greater) \\ [D^{-}, p^{-}] &= kstest(X_{contract}, X_{monitoring}, less) \end{aligned} \tag{6.7}$$

[4]Tradução de "One-sided two-sample Kolmogorov-Smirnov test"

para aceitar ou rejeitar H_0. Daí, H_0 é rejeitado quando o $p^+ < \alpha$ é pequeno, onde geralmente α é 0.05. Desse modo, detecta-se uma violação do contrato.

Por outro lado, para cumprir o contrato não é suficiente ter um valor $p^+ >= \alpha$, já que tem que se levar em consideração a flutuação aleatória de F'_S sobre F_S. Essa flutuação é medida por:

$$D = max\{D^+, D^-\}$$

Portanto, para que um conjunto de amostras $X_{monitoring}$ cumpra o contrato uma regra baseada em p^+ e D tem que ser verdadeira:

$$verify(X_{monitoring}) = \begin{cases} true, & \text{se } p^+ >= \alpha \wedge D^+ < \lambda \\ false, & \text{de outra maneira} \end{cases} \tag{6.8}$$

Onde a função $verify$ realiza a verificação do cumprimento do contrato das amostras $X_{monitoring}$, λ define a zona de tolerância da flutuação entre distribuições de probabilidade.

As principais características e motivos para usar o test de Kolmogorov-Smirnov (KS) são [SL11]:

- É um teste não-paramétrico, que compara as distribuições cumulativas de dois conjuntos de dados.
- Dado que o teste é não-paramétrico, não assume que os dados são coletados a partir de distribuições gaussianas (ou quaisquer outras distribuições definidas).
- Os resultados não mudam se transformam todos os valores para logaritmos ou recíprocos ou qualquer outra transformação. O teste de KS relata a diferença máxima entre duas distribuições cumulativas, e calcula um *p-value* a partir delas e dos tamanhos das amostras. Uma transformação vai esticar o eixo X da distribuição de freqüência, mas não pode alterar a distância máxima entre duas distribuições de freqüência.
- Porque testa para mais desvios a partir da hipótese nula do que outros testes como o *Mann-Whitney*. O teste de KS tem menos poder para detectar desvios na mediana, mas tem mais poder para detectar mudanças na forma da distribuição.
- Já que o teste não compara algúm parâmetro em particular (por exemplo, média ou mediana), o teste de KS não reporta intervalo de confiança.

6.7 Considerações Finais

Neste capítulo apresentaram-se as descrições das etapas que abrangem da definição de requisitos de QoS, estabelecimento do contrato de SLA, até o monitoramento com a detecção de violações de contrato em coreografias de serviços Web. Na definição de requisitos de QoS se propus usar um mapeamento de um subconjunto do total de elementos da especificação de coreografia em BPMN 2 para uma GSPN (Rede de Petri Estocástica Generalizada). Essa GSPN é uma representação intermediária da coreografia com suporte de QoS. O alvo é realizar avaliações de desempenho e a partir deles definir requisitos de QoS em etapas antecipadas como a modelagem.

Na definição de requisitos também foi proposto realizar simulações, mas dessa vez utilizando um simulador de coreografias com suporte de QoS chamado de *ChorSim*. Construiu-se um simulador de coreografias para alavancar a pesquisa de QoS em coreografias de serviços, já que coreografias de serviços é ainda uma linha de pesquisa emergente e há poucos trabalhos com relação a QoS, onde não existem infraestruturas maduras o suficiente para rodar coreografias. Além do mais, cabe ressaltar que o modelo de falhas não está descrito com detalhe, já que não é o foco e as falhas são utilizadas para definir cenários nas etapas da definição de requisitos e no monitoramento.

O contrato probabilístico para garantir a QoS é definido por meio de uma função de distribuição acumulada a partir dos valores estimados do atributo de QoS alvo utilizando o *ChorSim*. Na etapa de monitoramento descreveu-se a configuração do monitor e a técnica utilizada para detectar violações

de SLA, a qual está baseada no uso do teste de *Kolmogorov-Smirnov* de apenas um lado. Esse teste permite calcular quão próxima uma distruibuição está de outra e já que é de tipo não paramétrico podem ser testados diversos tipos de distribuições.

Capítulo 7

Resultados e Discussões

7.1 Cenários de Coreografia

Para atestar a eficácia da proposta, a coreografia apresentada em [BDF+08] é utilizada. Essa coreografia representa uma aplicação de CDN (*Content Delivery Network*) para fornecimento de conteúdo multimídia como audio, vídeo e imagens. A Figura 7.1 mostra o cenário de referência: um usuário requere, e eventualmente recebe, um serviço complexo gerido por meio de uma coreografia de diferentes serviços Web, um dos quais (WS_3 na figura) controla o provisionamento de conteúdo de *streaming*. WS_1 e WS_2 mostram que vários serviços Web são internamente orquestrados. A coreografia é composta de cinco serviços Web (WS_1,WS_2, WS_3, WS_4 e WS_5). Cada serviço pertence a um participante diferente e há sete canais de comunicação definindo a topologia G_{chor}, onde:

$G_{chor} = (V_{WS}, E)$ é um grafo não orientado, $WS_i \in V_{WS}$ é um serviço Web da coreografia e $e \in E$ é a comunicação entre dois serviços. $V_{WS} = \cup_{i=1}^{5}\{WS_i\}$ e $E = \{(WS_1, WS_2), (WS_1, WS_3), (WS_1, WS_4), (WS_2, WS_4), (WS_2, WS_5), (WS_3, WS_5)\}$.

Porém, nem sempre todos os serviços de uma coreografia são utilizados em uma instância de coreografia, isto é, há interações entre serviços que não acontecem porque a especificação de uma coreografia abrange várias possibilidades e todas as possíveis interações. Assim, os serviços utilizados em uma instância de coreografia dependem do ponto de entrada, ou seja, a primeira requisição para algum serviço da coreografia. Neste caso as instâncias de coreografias a simular, conforme a Figura 7.1, se iniciam com requisições do cliente para uma operação do serviço composto WS_1, que resulta em interações com os serviços WS_3 e WS_5.

7.2 Definição de Requisitos de QoS Analiticamente

Nesta seção todos os pasos da metodologia proposta para definir requsitos de QoS de maneira analítica são realizados. A coreografia apresentada na Figura 3.10 é utilizada. É um cenário simples para entendê-lo e o suficiente para mostrar a eficacia da metodologia proposta, já que inclui pelo menos um tipo dos três elemntos de BPMN de coreografias considerados (atividades, *gateways* e eventos). As próximas subseções desenvolvem cada um dos pasos da metodologia.

7.2.1 Mapeamento

A Figura 7.2 apresenta a GSPN obtida como resultado de realizar o mapeamento sobre a coreografia da Figura 3.10. As Tabelas 7.1 e 7.2 mostram a interpretação de todas as Posições e transições da GSPN resultante.As Posições representam o inicio e final da coreografia, os participantes, o inicio e fim do canal de comunicação nas interações, pontos de decisão e os diversos erros de acordo ao modelo de QoS.

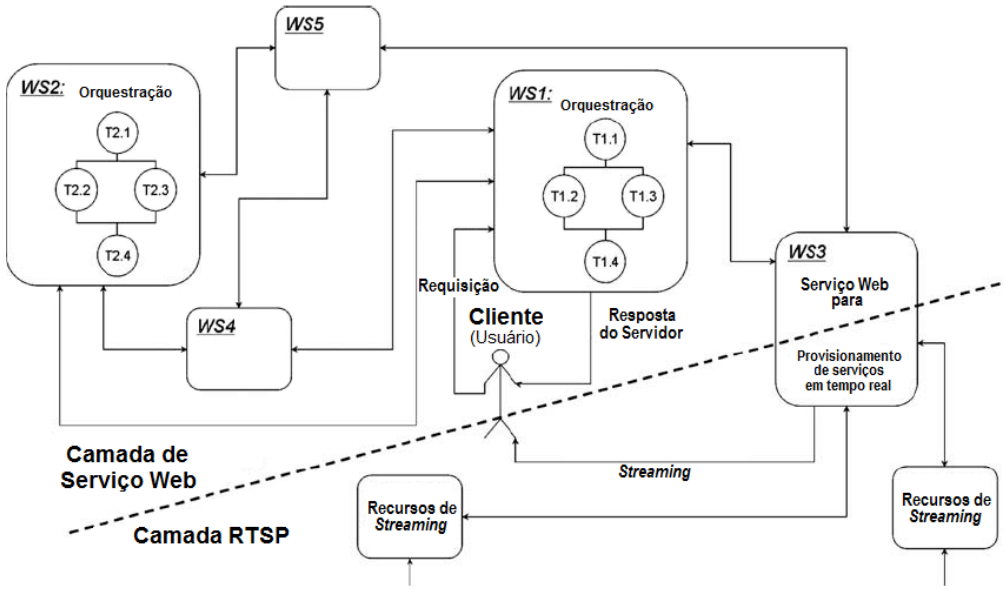

Figura 7.1: *Coreografia de serviços da aplicação de CDN [BDF+ 08]*

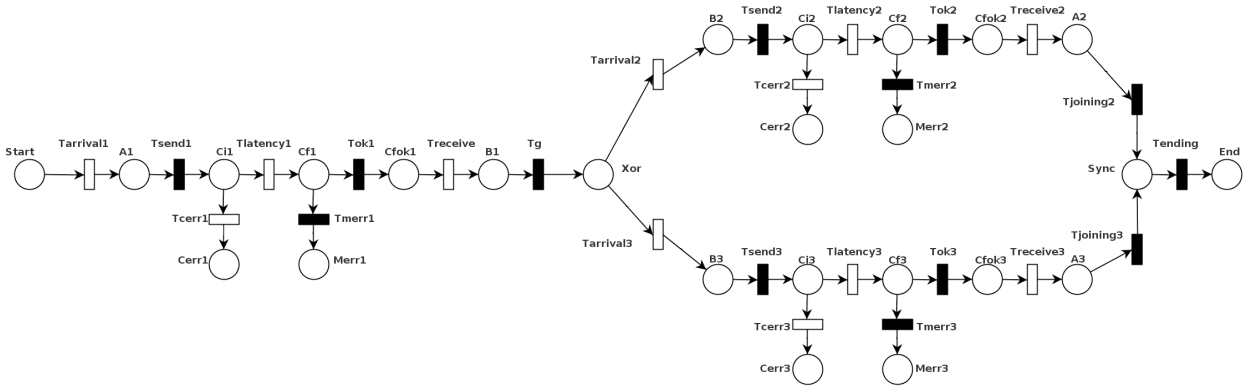

Figura 7.2: *GSPN obtido da coreografia do cenário de exemplo*

Tabela 7.1: *Descrição das posições da GSPN resultante*

Posição	Descrição
Start, *End*	Incio e fim da coreografia.
A_1, A_2, A_3	Representa ao participante "Cliente".
B_1, B_2, B_3	Representa ao participante "Consultor Financeiro".
C_{i1}, C_{i2}, C_{i3}	Inicio da comunicação.
C_{f1}, C_{f2}, C_{f3}	Fim da comunicação.
C_{fok1}, C_{fok2}, C_{fok3}	Formato de mensagem válido.
C_{err1}	Erro de comunicação entre A e B.
C_{err2}	Erro de comunicação entre B e A.
C_{err3}	Erro de comunicação entre B e A.
$M_{err1,\ 2,\ 3}$	Formato de mensagem inválido.
Xor e *Sync*	*Fork* e *Join* do *exclusive gateway XOR*.

7.2.2 Definição de pesos

O próximo passo é definir de pesos (*rates*) das transições de tempo a as prioridades nas transições immediatas. Esse valores dependem do conhecimento acerda do entorno onde as coreografias serão executadas e a confiabilidade dos serviços. A Tabela 7.3 apresenta a definição dos pesos e prioridades em dois cenários, os quais serão simulados no próximo passo. O cenário 1 está configurado para ter

Tabela 7.2: *Descrição de transições da GSPN resultante*

Transição	Descrição
$T_{arrival1}$	Chegada de instâncias de coreografia.
$T_{arrival2,\ 3}$	Chegada de instâncias de coreografia e decições de fluxo.
T_{send1}, T_{send2} e T_{send3}	Atividade e envio de mensagens.
$T_{latency1,\ 2,\ 3}$	Latência de rede.
$T_{receive,\ 2,\ 3}$	Recebimento de mensagem e execução do serviço.
$T_{cerr1,\ 2,\ 3}$	Erro de comunicação.
$T_{merr1,\ 2,\ 3}$	Falha no formato da mensagem.
T_g	Continuação do fluxo de sequência.
$T_{ok1}, T_{ok2}, T_{ok3}$	Mensagem válida recebida.
$T_{joining2,\ 3}$	Junção do fluxo.
T_{ending}	Final da instância de coreografia.

menor chance de falhas do que o cenário 2.

Tabela 7.3: *Configuração de pesos nos Cenários 1 e 2*

	Weights	
Transição	Cenário 1	Cenário 2
$T_{latency1}$, $T_{latency2}$, $T_{latency3}$	0.99	0.94
T_{cerr1}, T_{cerr2}, T_{cerr3}	0.01	0.06
$T_{receive}$, $T_{receive2}$, $T_{receive3}$	99	97
T_{merr1}, T_{merr2}, T_{merr3}	1	3
$T_{arrival2}$, $T_{arrival3}$	0.5	0.5

7.2.3 Simulações

Na simulação dos cenários, 100 *tokens* foram considerados para cada cenário na Posição de Inicio. Cada *token* representa uma instância de coreografia na simulação. Assim, para cada simulação, 100 instância concorrentes dos cenários foram executados (*multiple-server semantic*). As simulações dos dois cenários foram executados várias vezes de acordo com as configurações estabelecidas na Tabela 7.3. Foram realizados 1500 disparos com 10 replicações. A ferramenta Pipe2 [BPM+07] foi utilizada para modelar as GSPNs e simular os cenários.

A Tabela 7.4 mostra os resultados das simulações com um intervalo de confiabilidade de 95%. Os números apresentados na tabela mostram a média do número de *tokens* que cehgaram em cada uma das Posições da GSPN.

7.2.4 Análise de Resultados

Pelos resultados apresentados na Tabela 7.4 é possível notar que no Cenário 1, uma média de 1.52% ($C_{err1} + C_{err2} + C_{err3}$) das instâncias não finalizaram o processo devido a erros de redes no momento da troca de mensagens. Além disso, uma média de 0.39% ($M_{err1} + M_{err2} + M_{err3}$) das instâncias não finalizaram o processo devido a formatos de menssagens inválidos, que foram detectados quando as mensagens foram recebidas.

Similar ao cenário 1, no cenario 2, uma média de 3.10% e 2.50% das instâncias não finalizaram o processo devido a erros na rede formatos de mensagens inválidos respectivamente. Como esperado, os resultados estão de acordo com os parâmetros definidos para cada cenário (Tabela 7.3).

Um importante finding com as simulações desses cenários, e que pode ser útil para desenvolvedores e admnistradores de rede está relacioanda com a existência de gargalos na primeira interação (C_{i1}), a qual evitou que 8.32% e 8.90% das instâncias sejam enviadas pela rede. Essa informação

Tabela 7.4: *Resultados (em %)*

	Média do # de *tokens*		Intervalo de confiabilidade 95% (+/-)	
Place	Cenário 1	Cenário 2	Cenário 1	Cenário 2
Start	35.28	40.15	5.83	6.23
End	41.95	38.78	2.53	3.82
A_1	0.08	0.08	0.00	0.00
B_2	0.04	0.01	0.01	0.01
B_3	0.04	0.01	0.01	0.01
B_1	0.08	3.23	0.00	0.00
A_2	0.04	0.01	0.01	0.01
A_3	0.04	0.01	0.01	0.01
M_{err1}	0.39	0.91	0.95	1.92
M_{err2}	0.00	0.93	0.63	0.64
M_{err3}	0.00	0.66	0.87	0.74
C_{err1}	0.74	2.94	0.82	2.02
C_{err2}	0.00	0.00	0.67	1.75
C_{err3}	0.78	0.16	0.92	1.52
C_{i1}	8.32	8.90	5.33	7.48
C_{i2}	0.63	0.69	0.23	0.52
C_{i3}	0.75	8.90	0.39	0.21

pode afetar a modelagem e implementação coreografia definitiva, bem como definir novas políticas de QoS para serem configuradas na rede para reducir o atraso na execução.

7.3 Simulador de Coreografias

Como não havia nenhum sistema de simulador para coreografias de serviços, temos desenvolvido um simulador inicial para comparar o desempenho de um orquestrações e uma coreografia [GKB12]. Nesse trabalho, as coreografias revelaram ser uma melhor escolha respeito ao desempenho do que orquestrações. Porém, para os objetivos desta pesquisa se desenvolveu um simulador de coreografias com suporte de QoS, como especificado no capítulo anterior. A Figura 7.3 mostra o tempo de resposta médio, máximo e mínimo de um número de requisições simultâneas. Assim, as instâncias de coreografia envolveram requisições iniciais para WS_1, que pertence à cadeia de WS_1, WS_3 e WS_5 dependências de serviço do cenário descrito na Figura 7.1.

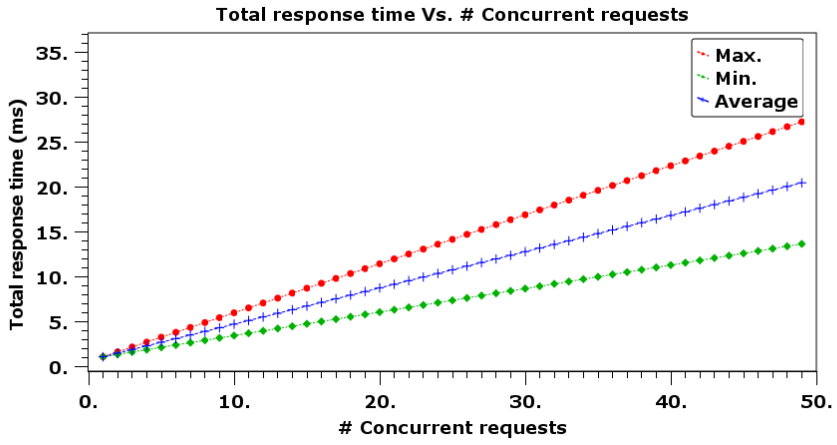

Figura 7.3: *Tempos de reposta médio, máximo e minimo de acordo ao número de requisições concorrentes.*

7.4 Definição de Requisitos de QoS em Coreografias

7.4.1 Metodologia dos Experimentos

O objetivo dos experimentos é analisar o comportamento do tempo de resposta total do serviço composto WS_1 em função do tamanho da resposta de WS_1 e diferentes valores de largura de banda. Para tanto, utilizam-se dois cenários definidos pelo comportamento da largura de banda da rede. O primeiro cenário possui valores de largura de banda que são fixos ao longo do tempo. O segundo cenário consiste em usar valores de largura de banda que são variáveis no decorrer do tempo representando assim a dinâmica do ambiente e a degradação por conta de falhas. A Figura 7.4 mostra como a largura de banda varia em um período de 100 segundos. Esta é a variação na simulação do segundo cenário. Essas variações foram geradas aleatoriamente.

A variável independente nos experimentos é o tamanho da resposta do serviço composto WS_1 que varia de $1KB$ até $100MB$. A variável dependente é o tempo médio da resposta total do serviço composto WS_1 de várias requisições simultâneas (que variam de 1 para 10 requisições) e de acordo à largura de banda da rede. Essa largura de banda está definida entre o canal de comunicação do Cliente e o serviço composto WS_1, e varia de 1Mbps até 16Mbps.

As Tabelas 7.6 e 7.7 apresentam os valores dos atributos de QoS das requisições e das respostas que são usados para as simulações. Cada simulação consistiu na execução de instâncias de coreografia iniciadas por requisições de um cliente. Desse modo, foram realizadas 960 simulações (1 a 5 requisições, 120 valores de tamanhos resposta e 16 valores de largura de banda) para cada cenário (modelo normal e variável da largura de banda). As simulações foram executadas em um computador equipado com processador Intel Core $i7 - 2700K$ 3.5Ghz, 16GB de memória RAM e 1TB de espaço em disco rodando o sistema operacional Debian GNU/Linux versão 6.0.

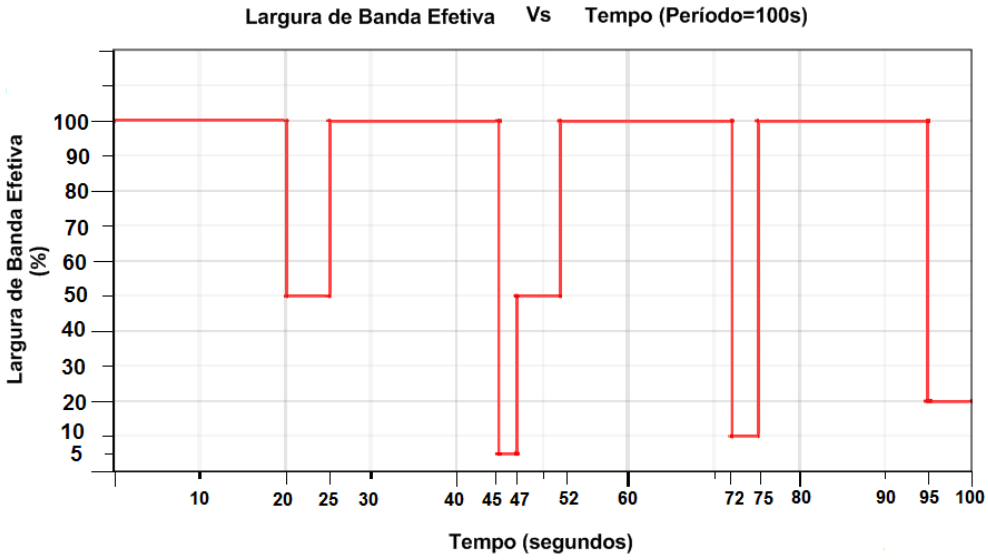

Figura 7.4: *Modelo de falhas que mostra a largura de banda efetiva devido à degradação da largura de banda referencial em um período de* 100 *segundos.*

Tabela 7.5: *Configuração dos cenários de simulação*

Atributo de QoS	Condição	Valor	Tipo de Falha
Tempo de Execução	fixo	Simulador	timeout= 100000ms (tolerância)
Vazão	variável	ws_i = 1 a 10 Requisições/ms	serviço não disponível = 0.5%
Formato da Mensagem	fixo	–	–
Latência	fixo	ws = 20ms,7ms, 5ms, 7ms, 5ms	Erro de comunicação= 3%

Tabela 7.6: *Configuração de valores dos atributos de QoS nas requisições*

Requisições	Largura de banda	Tamanho da requisição	latência	# requisições
Cliente a WS_1	1Mbps	1.95MB	0.002s	De 1 a 10
WS_1 a WS_3	1Mbps	5.47MB	0.002s	De 1 a 10
WS_3 a WS_5	1Mbps	5.47MB	0.002s	De 1 a 10

Tabela 7.7: *Configuração de valores dos atributos de QoS nas respostas*

Respostas	Largura de banda	Tamanho de resposta	latência	*timeout*
WS_1 a *Cliente*	1Mbps a 16Mbps	1KB a 100MB	0.002s	1000s
WS_3 a WS_1	20Mbps	8MB	0.002s	1000s
WS_5 a WS_3	40Mbps	200MB	0.002s	1000s

7.4.2 Análise de Resultados

Os gráficos das Figuras 7.5 e 7.6 mostram os resultados dos dois cenários de simulação, isto é, para o modelo normal e para o modelo variável com falhas. No modelo normal (Figura 7.5) percebe-se que o comportamento do tempo de resposta é menor quanto maior é a largura de banda, e estabiliza-se a partir de $5Mbps$ com um tempo de resposta de $150ms$. Além disso, com valores pequenos nos tamanhos de resposta de até $30MB$ os tempos de resposta são similares com larguras de banda a partir de $2Mbps$. Com base nisso já pode-se definir restrições de QoS tais como, poder oferecer tempos de resposta menores que $450ms$ desde que se tenha uma infraestrutura para suportar uma largura de banda de mais de $5Mbps$.

Já no modelo variável com falhas, os tempos de resposta diminuem quanto maior é a largura de banda, mas diferente do modelo normal, neste cenário começa a se estabilizar a partir de $14Mbps$ de largura de banda, apesar de a partir dos $8Mbps$ os tempos de resposta serem bem próximos. Além disso, o tempo de resposta tem quase o mesmo comportamento para todos os tamanhos de resposta, diferente do modelo normal, onde para tamanhos de resposta menores que $30MB$ podia se obter algumas vantagens. Neste cenário, houveram estouros de temporização (*timeouts*) com as larguras de banda de $1Mbps$ e $2Mbps$ com mais de 2 requisições simultâneas e por conta disso essas curvas não aparecem no gráfico. Dessa maneira, as restrições de QoS estariam focadas em garantir larguras de banda mínimas maiores que $2Mbps$ e para garantir tempos de resposta mínimos são necessários valores de largura de banda maiores que $14Mbps$.

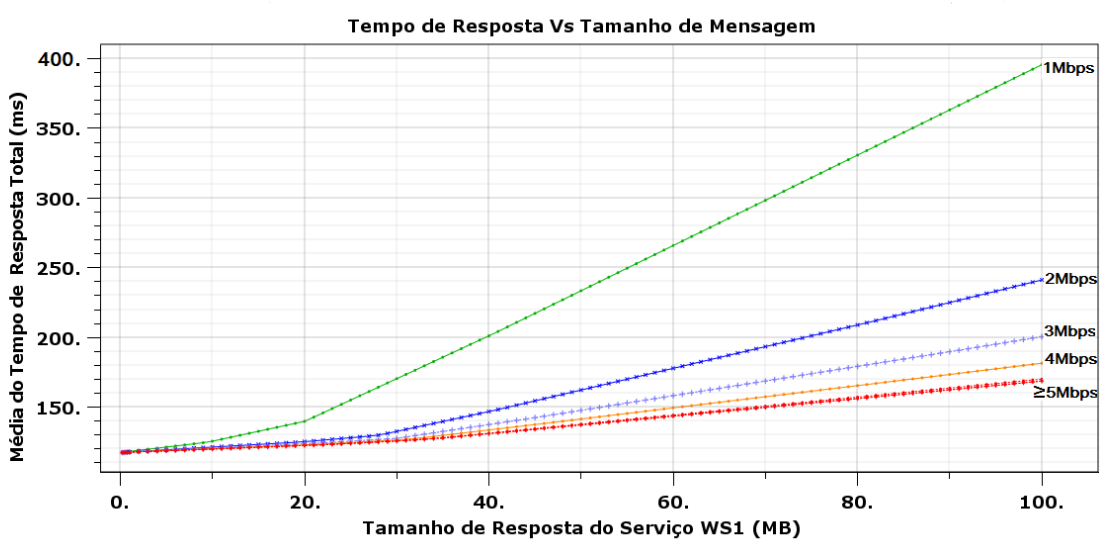

Figura 7.5: *Tempo médio de resposta total da coreografia em função do tamanho de resposta do serviço WS1 com larguras de banda de 1Mbps até 16Mbps (intervalos de confiança não são visíveis por terem ficado muito pequenos).*

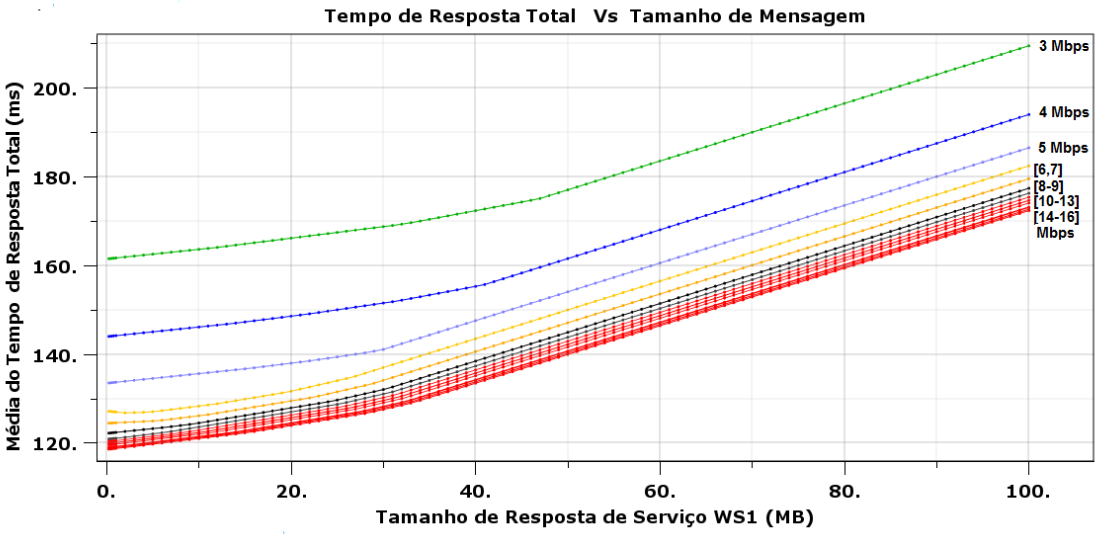

Figura 7.6: *Tempo médio de resposta total da coreografia em função do tamanho de resposta do serviço WS1 segundo o modelo de falha. A largura de banda varia de 1Mbps até 16Mbps*

7.5 Estabelecimento de contratos de QoS

Nesta etapa, vamos supor que a degradação do tempo de procesamento ou execução dos serviços individuais WS_1 e WS_3 e WS_5 se comportam de acordo com distribuições exponenciais. A escolha foi feita porque uma distribuição exponencial descreve o tempo para que um processo mude de estado, por isso é usado para modelar o comportamento de sistemas que tenham uma taxa de falha ou degradação no tempo. A Tabela 7.8 apresenta o tipo de distribuição de probabilidade usado e a taxa de degradação para os tempos de execução dos serviços WS_1 e WS_3 e WS_5. O resto da configuração é igual como é descrevido nas tabelas 7.7 e 7.6, exceto que a largura de banda nas repostas do WS_1 para cliente foi definida para ser de $10Mbps$ de acordo com a análise de resultados na definição de requerimentos de QoS.

Tabela 7.8: *Configuração das taxas de degradação dos serviços para obter o contrato do tempo de resposta para o serviço composto WS_1*

Serviço	Distribuição	Taxa de degradação (λ)
WS_1	Exponencial	1/10000
WS_3	Exponencial	1/10000
WS_5	Exponencial	1/10000

Desse modo, após obter por meio do ChorSim um conjunto de 5000 amostras dos tempos de reposta do serviço composto WS_1, já é possível obter a distribuição de probabilidade empírica (ECDF) F_{ws1} que represente o contrato. A Figura 7.7 mostra a ECDF obtido que representa o contrato probabilístico. Os quantis de F_{ws1} são mostrados também na Tabela 7.9.

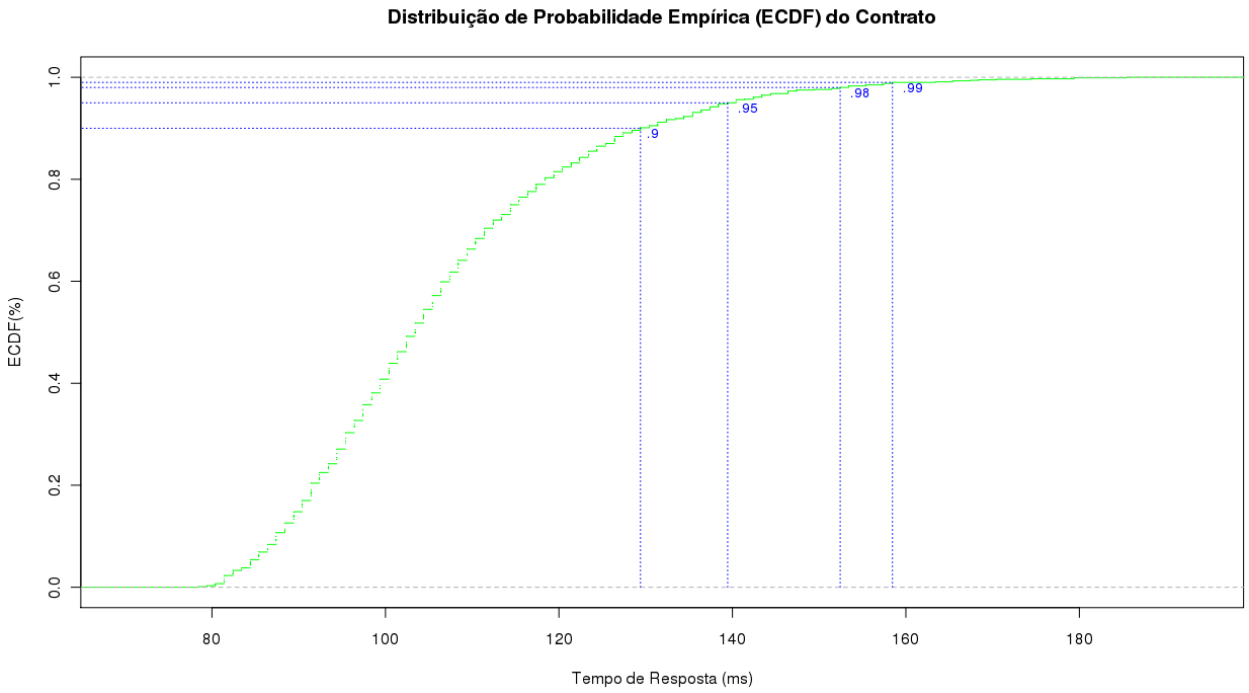

Figura 7.7: *Distribuição de probabilidade empírica (ECDF) do contrato do serviço WS_1 com base nos tempos de resposta*

Tabela 7.9: *Quantis do contrato probabilístico dos tempos de resposta de WS_1.*

Quantis	Tempo de Resposta
90%	129.4052 ms
95%	139.4552 ms
98%	152.4252 ms
98%	158.4552 ms

7.6 Monitoramento de QoS em Coreografias

Configuração

Além de ter estabelecido o contrato probabilístico e antes de começar o processo de monitoramento precisa-se configurar alguns outros parâmetros, tais como o tamanho da janela de amostras N, o desvio da janela de amostras p, e os parâmetros relacionados à regra de cumprimento de garantia de QoS.

A Tabela 7.10 mostra os valores utilizados para os parâmetros a usar no monitoramento. A regra específica para o cumprimento de garantia de QoS precisa definir os parâmetros de zona de tolerância λ e o nível de significância α. Desse modo, a regra de garantia fica da seguinte forma:

$$verify(X_{monitoring}) = \begin{cases} true, & \text{se } p^{+} >= 0.05 \wedge D^{+} < 0.15 \\ false, & \text{de outra maneira} \end{cases} \tag{7.1}$$

Onde $X_{monitoring}$ é o conjunto de amostras dos tempos de resposta do serviço composto WS_1. Para habilitar um monitoramento sequencial e *on-line* precisa-se computar um conjunto amostras sequenciais de tamanho N que que se sobrepõem. Assim, o conjunto de janelas de amostras a monitorar seria: $\{1, \ldots, N\}$, $\{p, \ldots, p+N\}$, $\ldots$, $\{mp, \ldots, mp+N\}$, e assim por diante, onede $p <= N$ e $m = 1, 2, \ldots$. O tamanho de amostras (N) para realizar a detecção é de 100, e o desvio p é 1.

Tabela 7.10: *Definição de valores no configuração para o monitoramento.*

Parâmetro	Valor
λ	0.15
α	0.05
N	100
p	1

Detecção de violações de SLA

Para mostrar a eficácia da abordagem de detecção de violações de SLA na etapa de monitoramento, desenvolveu-se diversos cenários de modo a apresentar diversos comportamentos nos tempos de reposta e na detecção de violações do contrato estabelecido. Tais cenários são caracterizados pela configuração de valores das taxas de degradação do tempo de execução dos serviços WS_1, WS_3 e WS_5. A Tabela 7.11 mostras os valores das taxas de degradação dos serviços Web para quatro cenários, onde do primeiro até o terceiro cenário possuem taxas de degradação maiores do que as degradações que foram usadas para o estebelecimento do contrato, mas que vão se aproximando como no terceiro cenário. O quarto cenário possui taxas de degradação de desempenho que são um pouco menores do que as usadas no estabelecimento contrato.

Tabela 7.11: *Estabelecimento das taxas de degradação de tempo de processamento dos serviços para os cenários.*

	Taxa de degradação (λ)		
Cenário	WS_1	WS_3	WS_5
Cenário 1	1/13000	1/12500	1/11500
Cenário 2	1/12000	1/11000	1/12000
Cenário 3	1/10500	1/10000	1/10500
Cenário 4	1/9000	1/10000	1/9000

Para cada um dos quatro cenários, o monitoramento se executou com simulações de 1000 requisições para o serviço composto WS_1. Além disso, a topologia e configuração da implantação (largura de banda, latência, capacidade de procesamento, etc) é a mesma como definida no estabelecimento de contrato probabilístico. Os apêndices A, B e C mostram com mais detalhes a especificação da coreografia, da plataforma e da implantação que o ChorSim precisa para rodar as simulações. Os gráficos 7.8, 7.9, 7.10 e 7.11 mostram as distâncias (D) e os p^+-*values* do teste de *Kolmogorov-Smirnov* utilizado na detecção de violações do contrato e ressalta-se os valores de λ e α com uma linha horizontal vermelha para cada um. Esses gráficos mostram a distância do teste de Kolmogorov, o qual mostra o quanto flutua a distribuição dos tempos de reposta respeito do contrato e de acordo com a regra (7.1) debe ser menor que 0.15 para ser cumprida. Os gráficos também mostram os p^+-*values* para cada uma das requisições realizadas, e de acordo com a regra em (7.1) devem ser maiores ou iguais que 0.05 para cumprir o contrato.

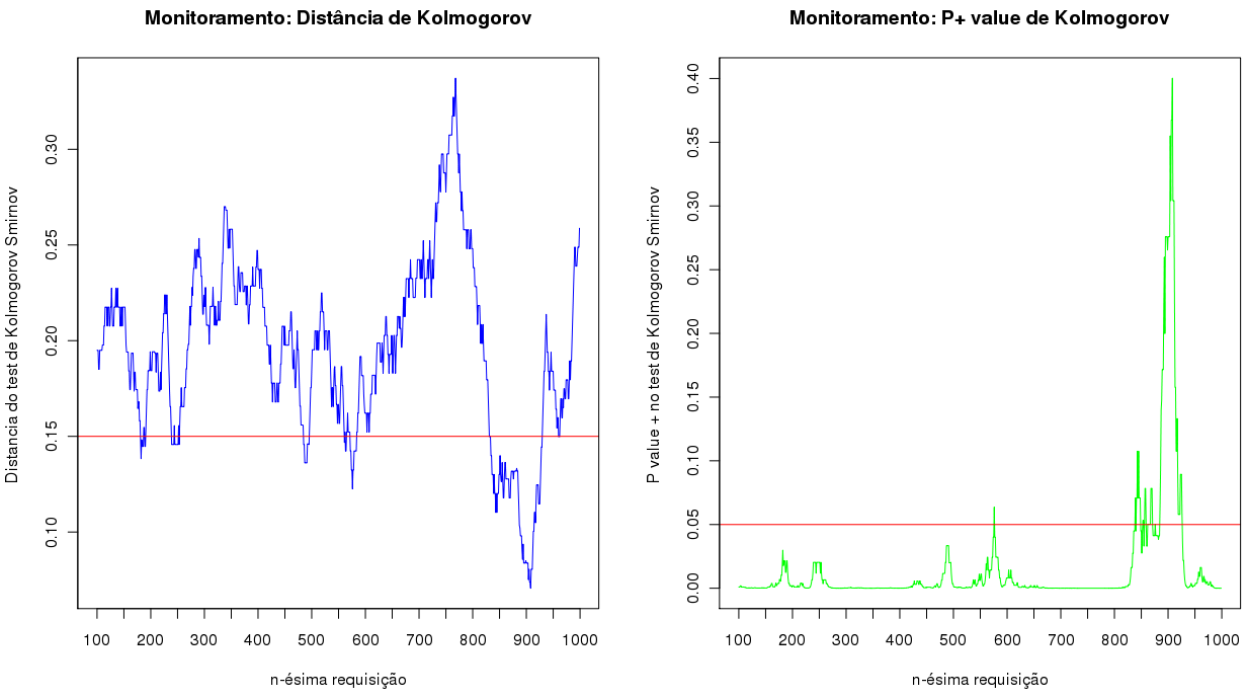

Figura 7.8: *Monitoramento e detecção de violações de SLA para o cenário 1.*

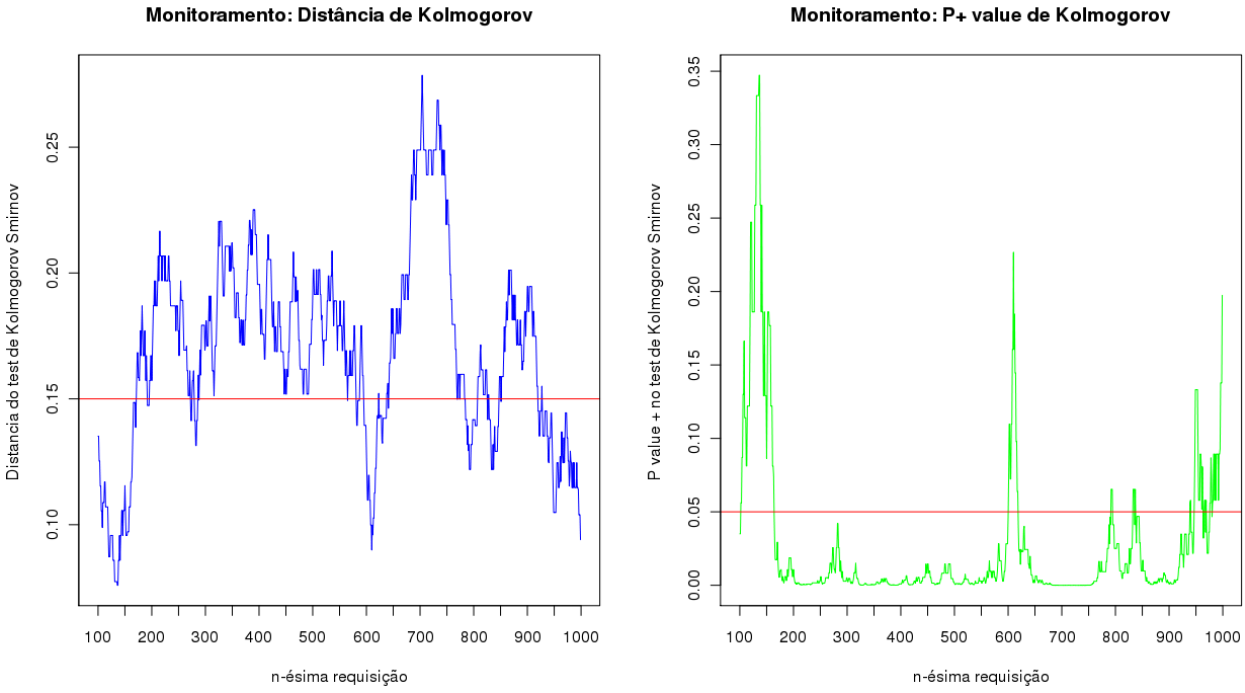

Figura 7.9: *Monitoramento e detecção de violações de SLA para o cenário 2*

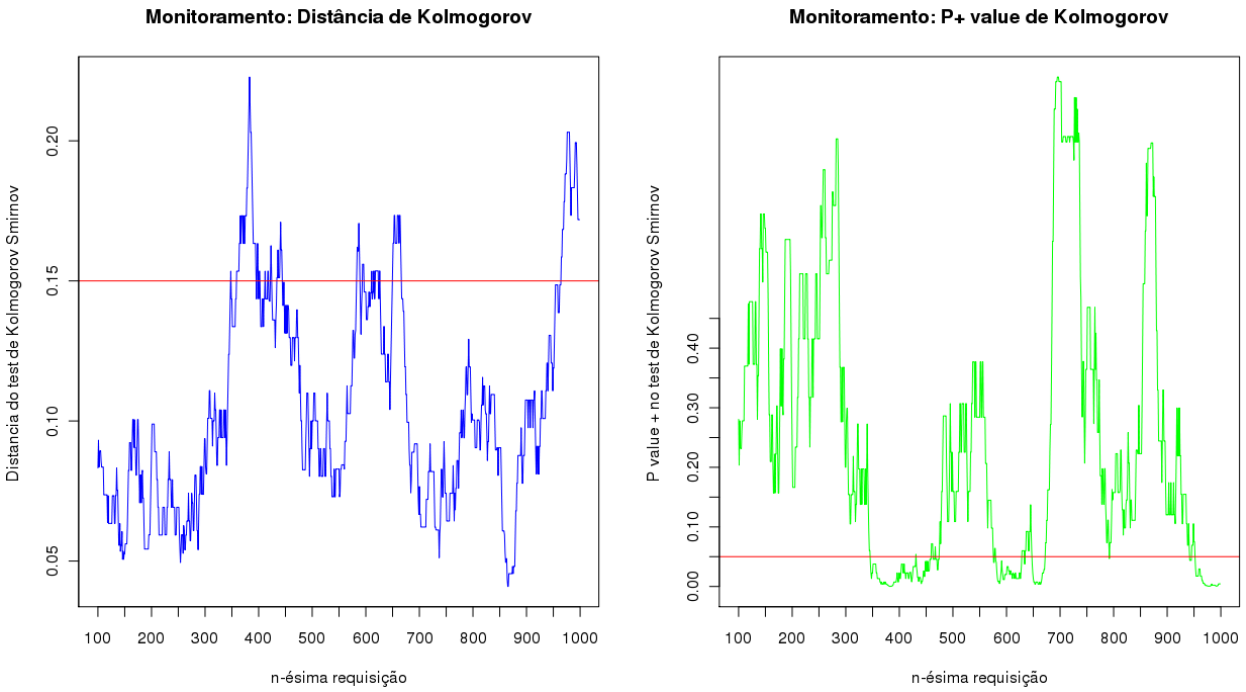

Figura 7.10: *Monitoramento e detecção de violações de SLA para o cenário 3*

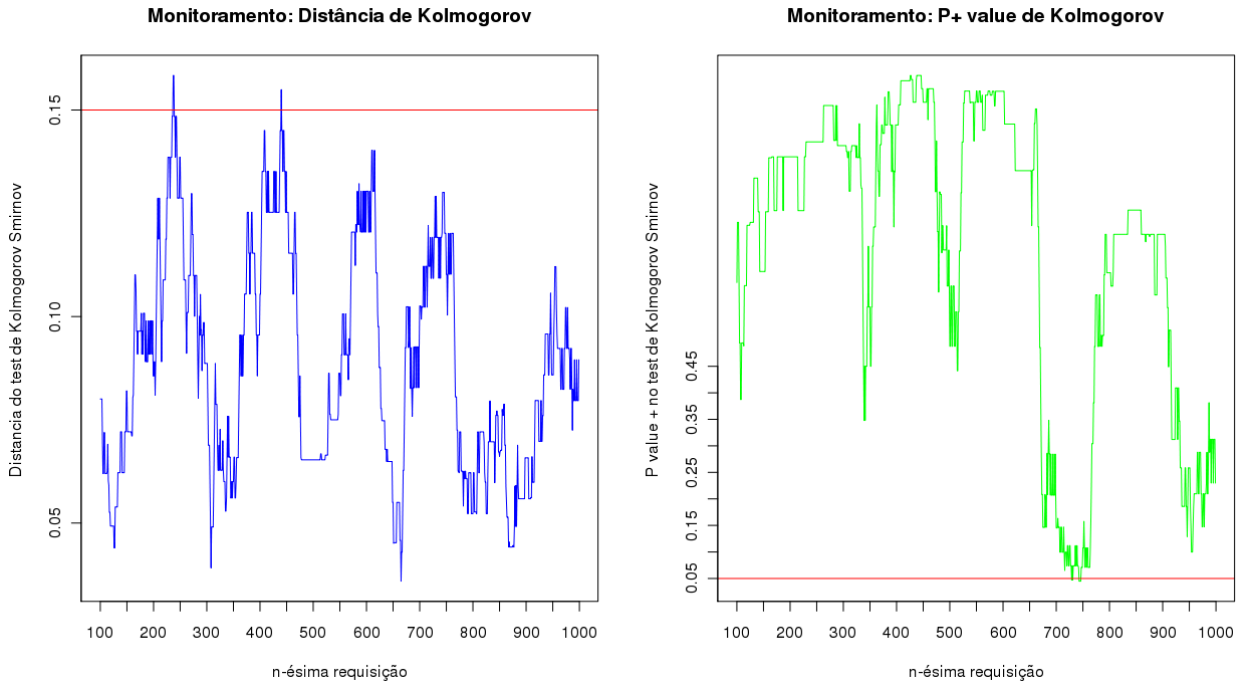

Figura 7.11: *Monitoramento e detecção de violações de SLA para o cenário 4*

No cenário 1 houve violações do contrato no 92.33% dos casos, o que se evidencia pelos vários *p-values* menores do que 0.05 e flutuações maiores do que a zona de tolerância λ definida. Apenas o contrato foi garantido no intervalo entre as requisições 839 e 926, onde as flutuações tem maior queda e os *p-values* atingem valores altos. No cenário 2 houve violações no 85.33% dos casos, onde a o contrato foi garantido poucas vezes e principalmente nos intervalos nas requisições de 103 a 163, de 601 a 618 e de 947 a 1000. Como no primeiro cenário, as requisições que cumpriram com o contrato coincidem em lugares onde há flutuações com picos baixos e *p-values* com picos altos.

A diferencia dos anteriores cenários, o cenário 3 apresenta violações de contrato no 28.22% dos casos. Neste caso as violações de contrato se correspondem nos intervalos onde há picos altos nas flutuações. Apesar que dessa vez os *p-values* que quebram o contrato estão próximos do 0.05, o que dá entender que as flutuações foram mais importantes para detectar as violações. Há menos contratos não cumpridos como se esperava já que as taxas de degradação dos serviços individuais são um pouco maiores das que foram utilizadas para obter o contrato.

No cenário 4 se detectaram violações de contrato no 1% das 1000 requisições. Dessa vez não houve *p-values* menores que 0.05, mas as flutuações foram determinates para detectar as poucas violações. No entanto, dado que neste cenário as taxas de degradação do tempo de processamento dos serviços é menor do que no contrato, não deviu-se detectar falhas mesmo sendo 1%. Então, para evitar esses falsos alarmes poderia se usar um valor λ (zona de tolerância) maior do que definido na regra atual. Por exemplo, poderiamos estabelecer $\lambda = 0.158$ porque é a maior distância achada das que quebraram o contrato.

7.7 Considerações Finais

Neste Capítulo apresentaram-se os experimentos e resultados das etapas que abrangem da definição de requisitos de QoS, estabelecimento do contrato de SLA e até o monitoramento com a detecção de violações de contrato em coreografias de serviços Web. Na definição de requisitos de Qos em coreografias apresentou-se os resultados dos experimentos por meio de simulações em uma GSPN com suporte de QoS. Desse modo, conseguiu-se achar falhas na comunicação, na integridade das mensagens e gargalhos em diversos pontos das interações entre serviços.

Na definição de requisitos também se realizaram experimentos por meio de simulações, que a diferencia de ferramentas analíticas como a GSPN se utilizou o simulador de coreografias *ChorSim*. O objetivo foi descobrir o comportamento do tempo de respota com diferentes tamanhos de mensagem e diferentes larguras de banda. Para tanto, definiram-se dois cenários baseados no comportamento da largura de banda, o primeiro com largura de banda uniforme e o segundo com comportamento váriável. A principal limitação no segundo cenário foi o impedimento de não poder especificar o comportamento da largura de banda por meio de distribuições de probabilidade, já que o *Simgrid* não suporta a definição de váriaveis aleatórias. Adicionar tal funcionalidade requereria mexer na implementação do *Simgrid*.

No entanto, o estabelecimento do contrato probabilístico foi possível porque se conseguiu definir o tempo de processamento por meio de distribuições de probabilidade, que já é suportado no *ChorSim*. Desse modo, no estabelecimento e no monitoramento usou-se taxas de degradação do desempenho por meio de distribuições exponenciais para definir o contrato e os diversos cenários a monitorar. A técnica para detectar violações de SLA está baseada no uso do teste de *Kolmogorov-Smirnov* de apenas um lado. Esse teste permite calcular quão próxima uma distruibuição está de outra e já que é de tipos não paramétrico podem ser testados diversos tipo de distribuições.

Capítulo 8

Conclusões

Neste Capítulo final da dissertação, são apresentadas as principais contribuições do trabalho e sugestões para futuros trabalhos acerca desta pesquisa.

8.1 Considerações Finais

A presente pesquisa apresentou diversas abordagens e técnicas das etapas que abrangem da definição de requisitos de QoS, estabelecimento do contrato de SLA, até o monitoramento com a detecção de violações de contrato em coreografias de serviços Web, que é o objetivo principal. No desenvolvimento de todas as etapas se levou em consideração o modelo de interação de coreografias e padrão BMPN 2 que o suporta.

Na definição de requisitos de QoS em coreografias apresentou-se uma nova metodologia para avaliar o impacto da QoS por meio de simulações em uma GSPN. A metodologia inclui um algoritmo para mapear uma especificação de coreografia para uma GSPN que possui características para QoS. Desse modo, conseguiu-se achar falhas na comunicação, na integridade das mensagens e gargalhos em diversos pontos das interações entre serviços em etapas antecipadas como a modelagem.

Porém, os objetivos foram atingidos por meio de simulações de coreografias, mas a diferencia de usar uma ferramentas analíticas como a GSPN, construiu-se um simulador de coreografias com suporte de QoS chamado de *ChorSim*. A construção do simulador leva em consideração elementos do modelo de interação de coreografias (atividades de coreografia, *gateways*, entre outros) que é oferecido no padrão BPMN2. O simulador permite rodar coreografias, realizar medições de atributos de QoS atômicos e compostos, e possui um monitoramento para acompanhar a execução das instâncias de coreografias.

Dessa maneira, é possível simular comportamentos que atualmente não podem ser realizados pela falta de implementações maduras e que permitam realizar pesquisas em coreografias de serviços Web, especialmente as relacionadas com QoS e monitoramento. Atualmente o simulador de coreografias já suporta o estabelecimento do tempo de procesamento nos serviços por meio de distribuições de probabilidade. Porém, os atributos de QoS relacionados com comunicação (largura de banda, latência, entre outros) ainda não possuem essa característica porque o *Simgrid* não a suporta.

Assim, a definição de requisitos de QoS se atingiu com base nos resultados das simulações usando *ChorSim* usando cenários com diferentes comportamento na largura de banda e observando seu impacto no tempo de resposta total. A partir dos resultados das simulações se estabelecer requisitos e restrições de QoS de maneira a definir SLAs iniciais.

O estabelecimento do contrato probabilístico foi possível porque se conseguiu definir o tempo de processamento por meio de distribuições de probabilidade. No estabelecimento do contrato probabilístico e no monitoramento para detecção de violações de SLA se utilizaram taxas de degradação do desempenho por meio de distribuições exponenciais. Na etapa de monitoramento a configuração de diversas taxas de degradação permitiram definir cenários para mostrar a eficácia da abordagem de detecção de violações de contrato de QoS. Assim, a detecção de violações de SLA está baseada no uso do teste de *Kolmogorov-Smirnov* de apenas um lado, o qual permite calcular quão próxima

uma distruibuição qualquer está de outra. Os resultados nessa última etapa foram consistentes com as taxas de degradação definidas para os diversos cenários.

Finalmente, os resultados não foram comparados com outras propostas já que coreografias de serviços Web é ainda uma linha de pesquisa emergente e há poucos trabalhos com relação a QoS. Não existem infraestruturas maduras o suficiente para rodar coreografias e, em consequência, ainda não dá para validar os resultados das simulações com implementações reais.

8.2 Trabalhos Futuros

Esta pesquisa está focada em atributos QoS não determinísticos de desempenho e confiabilidade. Futuros trabalhos podem adicionar outros tipos, tais como o custo, pagamento, segurança, reputação, entre outros. Além disso, o monitoramento poderia incluir predição de QoS baseado no histórico dos valores medidos anteriormente, isto serve de base para realizar, por exemplo, auto-cura (*self-healing*) em vários níveis (serviços, orquestração e coreografia), melhorar a adaptação, etc.

Nesta pesquisa somente foram levados em consideração um subconjunto do total de elementos disponíveis para especificar coreografias com o padrão BPMN 2. Trabalhos futuros deveriam envolver suporte de mais elementos de coreografias de modo a poder simular completamente uma coreografia de processos.

Dado que o monitoramento baseado em QoS e SLA é a um fator chave para alavancar a adaptação, então trabalhos futuros podem considerar usar o monitoramento como base para uma ferramenta que forneça de adaptação dinâmica, reconfiguração e autocura baseada em QoS para coreografias de serviços Web.

Atualmente dá para especificar degradações do tempo de processamento dos serviços individuais por meio de distribuições de probabilidade. O suporte para realizar o mesmo com atributos de comunicação (largura de banda, latência, etc) precisará de adicionar funcionalidade no *core* do *Simgrid* mesmo.

Atualmente, no projeto *CHOReOS* está se implementando um motor de execução de coreografias, o que habilitaria a posibilidade de realizar validações dos resultados obtidos com o simulador. No futuro, estamos considerando utilizar diversos exemplos mais complexos de modo a a melhorar e sxtender as abordagens propostas nesta pesquisa.

Apêndice A

Especificação da Coreografia para o monitoramento no ChorSim

```
<?xml version='1.0'?>
 <choreography name="CDN application">
  <services>
    <service name="WS1" >
      <serviceOperation name="method1" />
      <serviceOperation name="method2" />
    </service>
    <service name="WS2" >
      <serviceOperation name="method1" />
      <serviceOperation name="method2" />
      <serviceOperation name="method3" />
    </service>
    <service name="WS3" >
      <serviceOperation name="method1" />
      <serviceOperation name="method2" />
    </service>
    <service name="WS4" >
      <serviceOperation name="method1" />
      <serviceOperation name="method2" />
      <serviceOperation name="method3" />
    </service>
    <service name="WS5" >
      <serviceOperation name="method1" />
      <serviceOperation name="method2" />
    </service>
  </services>
  <interactions>
    <interaction name="interactionWS1-WS3" >
      <source service="WS1" operation="method1" />
      <target service="WS3" operation="method2" type="REQUEST_RESPONSE" />
    </interaction>
    <!-- <interaction name="interactionWS1-WS3_WS4" >
      <source service="WS3" operation="method2" />
      <gateway type="parallel" />
      <target service="WS5" name="method1" type="REQUEST_RESPONSE" />
      <target service="WS4" name="method2" type="REQUEST" />
      <gateway type="join" />
    </interaction> -->
    <interaction name="interactionWS2-WS4" >
      <source service="WS2" operation="method1" />
      <target service="WS4" operation="method3" type="REQUEST_RESPONSE" />
    </interaction>
```

```
43    <!--<interaction name="interactionWS5-WS2" >
44      <source service="WS5" operation="method1" />
45      <target service="WS2" operation="method3" type="REQUEST_RESPONSE" />
46    </interaction> -->
47    <!-- <interaction name="interactionWS1-WS2" >
48      <source service="WS1" operation="method1" />
49      <target service="WS2" operation="method3" type="REQUEST_RESPONSE" />
50    </interaction> -->
51    <interaction name="interactionWS3-WS5" >
52      <source service="WS3" operation="method2" />
53      <target service="WS5" operation="method2" type="REQUEST_RESPONSE" />
54    </interaction>
55    <!--<interaction name="interactionWS5-WS1" >
56      <source service="WS5" operation="method2" />
57      <target service="WS1" operation="method2" type="REQUEST" />
58    </interaction> -->
59  </interactions>
60 </choreography>
```

Apêndice B

Configuração da plataforma para o monitoramento no ChorSim

```
1  <?xml version='1.0'?>
2  <!DOCTYPE platform SYSTEM "http://simgrid.gforge.inria.fr/simgrid.dtd">
3  <platform version="3">
4  <AS id="AS0" routing="Full">
5    <host id="Jacquelin" power="137333000"/>
6    <host id="iRMX" power="68667000"/>
7    <host id="Revoada" power="68667000"/>
8    <host id="Bellemarre" power="34333000"/>
9    <host id="Vincent" power="34333000"/>
10   <host id="Dali" power="34333000"/>
11
12   <link id="100" bandwidth="100000" latency="0.02"/>
13   <link id="1000" bandwidth="1000000" latency="0.02"/> -->
14
15   <link id="100Kbps" bandwidth="100000" latency="0.002"/>
16   <link id="1Mbps" bandwidth="1000000" latency="0.002"/>
17   <link id="3Mbps" bandwidth="3000000" latency="0.002"/>
18   <link id="5Mbps" bandwidth="5000000" latency="0.002"/>
19   <link id="10Mbps" bandwidth="10000000" latency="0.002"/>
20   <link id="20Mbps" bandwidth="20000000" latency="0.002"/>
21   <link id="40Mbps" bandwidth="40000000" latency="0.002"/>
22
23
24   <route src="Jacquelin" dst="Bellemarre" symmetrical="NO">
25      <link_ctn id="1Mbps"/>
26   </route>
27   <route src="Bellemarre" dst="Jacquelin" symmetrical="NO">
28      <link_ctn id="10Mbps"/>
29   </route>
30
31
32   <route src="Bellemarre" dst="Revoada" symmetrical="NO">
33      <link_ctn id="1Mbps"/>
34   </route>
35   <route src="Revoada" dst="Bellemarre" symmetrical="NO">
36      <link_ctn id="20Mbps"/>
37   </route>
38
39
40   <route src="Revoada" dst="Vincent" symmetrical="NO">
41      <link_ctn id="1Mbps"/>
42   </route>
```

```
43  <route src="Vincent" dst="Revoada" symmetrical="NO">
44     <link_ctn id="40Mbps"/>
45  </route>
46
47     <!-- Finalizing tasks from Jacquelin-->
48  <route src="Jacquelin" dst="Revoada" symmetrical="NO">
49     <link_ctn id="100Kbps"/>
50  </route>
51  <route src="Jacquelin" dst="Vincent" symmetrical="NO">
52     <link_ctn id="100Kbps"/>
53  </route>
54
55     </AS>
56     </platform>
```

Apêndice C

Configuração da mmplantação para o monitoramento no ChorSim

```
<?xml version='1.0'?>
<!DOCTYPE platform SYSTEM "http://simgrid.gforge.inria.fr/simgrid.dtd">
<platform version="3">

  <process host="Bellemarre"
  function="br.usp.ime.simulation.datatypes.process.Service">

     <argument value="50"/><!-- number of threads-->
     <argument value="WS1"/><!--service name-->
     <argument value="9000"/><!-- degradation mean (1/rate) (ms) -->

     <argument value="method"/><!--method-->
    <argument value="method1"/><!-- method name -->
    <argument value="7000000" /> <!-- Method computing amount -->
    <argument value="100663296" /> <!-- Default response Size 12MB-->

    <argument value="dependency"/><!-- method exec time-->
     <argument value="WS3"/><!-- service name-->
     <argument value="method2"/><!-- operation name-->
     <argument value="7000000" /> <!-- Method computing amount -->
     <argument value="5734400" /> <!-- Request Size 700KB-->

     <argument value="method"/><!--method-->
    <argument value="method2"/><!-- method name -->
    <argument value="7000000" /> <!-- Method computing amount -->
    <argument value="200000" /> <!-- Default response size -->
  </process>



  <process host="Revoada"
  function="br.usp.ime.simulation.datatypes.process.Service">

     <argument value="50"/><!-- number of threads-->
     <argument value="WS3"/><!--service name-->
     <argument value="10000"/><!-- degradation mean (1/rate) (ms) -->

     <argument value="method"/><!--method-->
    <argument value="method1"/><!-- method name -->
    <argument value="7000000" /> <!-- Method computing size -->
    <argument value="3800" /> <!-- Default Response size -->

```

```
41      <argument value="method"/><!--method-->
42    <argument value="method2"/><!-- method name -->
43    <argument value="7000000" /> <!-- Method computing size -->
44    <argument value="8388608" /> <!-- Default Response Size 1MB-->
45    <argument value="dependency"/><!-- method exec time-->
46      <argument value="WS5"/><!-- service name-->
47      <argument value="method2"/><!-- operation name-->
48      <argument value="3000000" /> <!-- Method computing amount -->
49      <argument value="5734400" /> <!-- Request size 700KB-->
50
51  </process>
52
53  <process host="Vincent"
    function="br.usp.ime.simulation.datatypes.process.Service">
54
55      <argument value="50"/><!-- number of threads-->
56      <argument value="WS5"/><!--service name-->
57      <argument value="9000"/><!-- degradation mean (1/rate) (ms) -->
58
59      <argument value="method"/><!--method-->
60    <argument value="method1"/><!-- method name -->
61    <argument value="7000000" /> <!-- Method compute duration (flops) -->
62    <argument value="20000" /> <!-- Default response size -->
63
64      <argument value="method"/><!--method-->
65    <argument value="method2"/><!-- method name -->
66    <argument value="7000000" /> <!-- Method computing size -->
67    <argument value="209715200" /> <!-- Response Size 25MB-->
68
69  </process>
70
71  <process host="Jacquelin"
72    function="br.usp.ime.simulation.choreography.Choreographer">
73
74    <!-- <argument value="1000" /> Amount of choreography requests
      (instances) -->
75    <argument value="0" /> <!-- Requests per second -->
76
77    <argument value="WS1" /> <!-- name of service to invoke-->
78    <argument value="method1"/><!-- name of method of service -->
79    <argument value="2048000"/><!-- Request size 250KB-->
80    <argument value="10" /> <!-- Size of each returned value -->
81  </process>
82
83 </platform>
```

Referências Bibliográficas

[ABPRT04] Daniel Austin, Abbie Barbir, Ed Peters, e Steve Ross-Talbot. Web Service Choreography Requirements, 2004. 45, 47

[ACKM04] Gustavo Alonso, Fabio Casatu, Harumi Kuno, e Vijay Machiraju. *Web Services: Concepts, Architecture and Applications*. Springer-Verlag, 2004. 6

[ADHR05] W M P Van Der Aalst, M Dumas, A H M Hofstede, e N Russell. Life After BPEL ? Em *Proceedings of the Formal Techniques for Computer Systems and Business Processes*, volume 3670, páginas 35—-50. Springer-Verlag, 2005. viii, 17, 18, 37

[BA11] Khoutir Bouchbout e Zaia Alimazighi. Inter-Organizational Business Processes Modelling Framework. Em *Advanced in Databases and Information Systems ADBIS 2011*, 2011. 19

[BBRW09] Adam Barker, Paolo Besana, David Robertson, e Jon B Weissman. The Benefits of Service Choreography for Data-Intensive Computing. Em *In Proceedings of the 7th international workshop on Challenges of large applications in distributed environments*, CLADE '09, páginas 1–10, 2009. 31, 55, 56

[BCdFZ10] Daniel M Batista, Luciano Chaves, Nelson L. S. da Fonseca, e Artur Ziviani. Performance Analysis of Available Bandwidth Estimation Tools for Grid Networks. *The Journal of Supercomputing*, 53(1):103—-121, 2010. 1

[BCG+05] Daniela Berardi, Diego Calvanese, Giuseppe De Giacomo, Richard Hull, e Massimo Mecella. Automatic Composition of Transition-based Semantic Web Services with Messaging. *31st Intl. Conference on Very Large Databases*, páginas 613—-624, 2005. 22

[BDF+08] F Buccafurri, P Demeo, M Fugini, R Furnari, a Goy, G Lax, P Lops, S Modafferi, B Pernici, e D Redavid. Analysis of QoS in cooperative services for real time applications. *Data & Knowledge Engineering*, 67(3):463–484, Dezembro 2008. viii, ix, 34, 35, 64, 65

[BDH05] Alistair Barros, Marlon Dumas, e Arthur H M Hofstede. Service Interaction Patterns. Em *Proceeedings of 3rd International Conference on Business Process Management*, páginas 302–318, Nancy, France, 2005. Springer Verlag. 23, 27, 61

[BDO05] A Barros, Marlon Dumas, e P Oaks. A critical overview of the web services choreography description language, 2005. 17, 24

[BDO06] Alistair Barros, Marlon Dumas, e Phillipa Oaks. Standards for Web Service Choreography and Orchestration : Status and Perspectives. Em Christoph Bussler Haller e Armin, editors, *Order A Journal On The Theory Of Ordered Sets And Its Applications*, volume 3812 of *Lecture Notes in Computer Science*, páginas 61–74. Springer Berlin / Heidelberg, 2006. 19

[BF08] Tevfik Bultan e Xiang Fu. Specification of realizable service conversations using collaboration diagrams. *IEEE International Conference on Service-Oriented Computing and Applications (SOCA)*, 2(1):27–39, Abril 2008. 22

[BFHS03] Tevfik Bultan, Xiang Fu, Richard Hull, e Jianwen Su. Conversation Specification: A New Approach to Design and Analysis of E-Service Composition. *12th international conference on World Wide Web (2003)*, páginas 403–410, 2003. 22

[BMCC02] Rajkumar Buyya, Manzur Murshed, Caulfield Campus, e Gippsland Campus. GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing. *CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE (CCPE)*, 14:1175–1220, 2002. 57

[BPB+09] Paolo Besana, Vivek Patkar, Adam Barker, David Robertson, e David Glasspool. Sharing Choreographies in OpenKnowledge: A Novel Approach to Interoperability. *Journal of Software (JSW)*, 4(8):833–842, Outubro 2009. 24

[BPM+07] Pere Bonet, Ramon Puigjaner, Palma De Mallorca, William J Knottenbelt, South Kensington Campus, e United Kingdom. PIPE v2.5: a Petri Net Tool for Performance Modeling. Relatório técnico, 2007. 66

[BR09] Tevfik Bultan e Nima Roohi. Realizability of Choreographies using Process Algebra Encodings. Em *Proceedings of the 7th International Conference on Integrated Formal Methods*, number 99 in IFM 2009. LNCS 5423, páginas 1–14. Springer-Verlag Berlin, Heidelberg, 2009. 21

[Bul08] Tevfik Bultan. Service Choreography and Orchestration with Conversations. Em *Proceedings of the 19th international conference on Concurrency Theory*, CONCUR '08, páginas 10–11. Springer-Verlag Berlin, Heidelberg, 2008. 21

[BWR09] Adam Barker, Christopher D. Walton, e David Robertson. Choreographing Web Services. *IEEE Transactions on Services Computing*, 2(2):152–166, 2009. viii, 1, 23, 32

[CCMW01] Erik Christensen, Francisco Curbera, Greg Meredith, e Sanjiva Weerawarana. W3C Web Services Description Language (WSDL) 1.1, 2001. 7

[CDMV09] María Emilia Cambronero, Gregorio De, Enrique Martnez, e Valentn Valero. A Comparative Study between WSCI, WS-CDL, and OWL-S. *Proceeedings of the IEEE International Conference on e-Business Engineering*, páginas 377–382, 2009. 17

[CHY07] Marco Carbone, Kohei Honda, e Nobuko Yoshida. Structured Communication-Centred Programming for Web Services. Em *Proceedings of the 16th European conference on Programming*, Lecture Notes in Computer Science, páginas 1–16, Berlin, Heidelberg, 2007. Springer-Verlag. 17

[CLQ08] Henri Casanova, Arnaud Legrand, e Martin Quinson. SimGrid: a Generic Framework for Large-Scale Distributed Experiments. Em *Proceedings of the Tenth International Conference on Computer Modeling and Simulation*, UKSIM '08, páginas 126–131. IEEE Computer Society, 2008. 57, 58

[CRBS98] E. Crawley, R.Nair, B. Rajagopolan, e H. Sandick. RFC 2386 - A Framework for QoS-based Routing in the Internet, 1998. 36

[DB08] Gero Decker e Alistair Barros. Interaction Modeling using BPMN. Em *In Proceedings of the 2007 international conference on Business process management (BPM'07)*, páginas 208–219. Springer-Verlag, Berlin, Heidelberg, 2008. 27

[DDO07] Remco M Dijkman, Marlon Dumas, e Chun Ouyang. Formal Semantics and Analysis of BPMN Process Models using Petri Nets. Relatório técnico, Australian Research Council under Discovery Grant DP0451092, 2007. viii, 25, 54

[DKB08] Gero Decker, Oliver Kopp, e Alistair Barros. An Introduction to Service Choreographies. *Information Technology*, 50(2):122–127, Fevereiro 2008. 23, 24

[DKLW07] Gero Decker, Oliver Kopp, Frank Leymann, e Mathias Weske. BPEL4Chor: Extending BPEL for Modeling Choreographies. Em *Proceedings of the IEEE International Conference on Web Services (ICWS 2007)*, páginas 296–303, Utah-USA, 2007. IEEE Computer Society. 18, 24

[Dou05] Andrew Douglas. WS-CDL Eclipse, 2005. 24

[DP07] Gero Decker e Frank Puhlmann. Extending BPMN for Modeling Complex Choreographies. Em *Proceedings of the 2007 OTM Confederated international conference on On the move to meaningful internet systems: CoopIS, DOA, ODBASE, GADA, and IS*, OTM'07, páginas 24—-40. Springer-Verlag, 2007. 27

[EA06] Mauricio Espinoza e Pedro Alvarez. A life cycle of a composite Web service: strengths and weaknesses of current solutions. Relatório técnico, Department of Engineering and Informatic Systems, University of Zaragoza, 2006. viii, 12

[Eng09] Lasse Engler. *BPEL gold: Choreography on the Service Bus*. Thesis, University of Stuttgart, 2009. viii, 23

[FPR09] Maria Grazia Fugini, Barbara Pernici, e Filippo Ramoni. Quality analysis of composed services through fault injection. *Vital And Health Statistics. Series 20 Data From The National Vitalstatistics System Vital Health Stat 20 Data Natl Vital Sta*, (May 2008):227–239, 2009. 59

[FUMK06] Howard Foster, Sebastian Uchitel, Jeff Magee, e Jeff Kramer. LTSA-WS: A Tool for Model-Based Verification of Web Service Compositions and Choreography. Em *Proceedings of the 28th International Conference on Software Engineering*, ICSE '06, páginas 771—-774, New York - USA, 2006. ACM. 24

[GG11] Nawal Guermouche e France Claude Godart. Characterizing Compatibility of Timed Choreography. *International Journal of Web Services Research*, 8(June):2008–2010, 2011. 22

[GHJV94] Erich Gamma, Richard Helm, Ralph Johnson, e John M. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1994. 9

[GKB12] Felipe Pontes Guimaraes, Eduardo Hideo Kuroda, e Daniel Macedo Batista. Performance Evaluation of Choreographies and Orchestrations with a New Simulator for Service Compositions. Em *Workshop on Computer-Aided Modeling Analysis and Design of Communication Links and Networks (CAMAD)*, páginas 140–144, 2012. 57, 67

[GZ10] Chunhua Gu e Xueqin Zhang. An SOA Based Enterprise Application Integration Approach. *2010 Third International Symposium on Electronic Commerce and Security*, páginas 324–327, Julho 2010. 10

[HGDJ08] Riadh Ben Halima, Karim Guennoun, Khalil Drira, e Mohamed Jmaiel. Non-intrusive QoS monitoring and analysis for self-healing web services. *Proceedings of the 1st International Conference on the Applications of Digital Information and Web Technologies (ICADIWT)*, (1):549–554, 2008. 36, 46

[Hil09] Marc Oriol Hilari. *Quality of Service (QoS) in SOA Systems: A Systematic Review.* Tese de Doutorado, Catalunya University, 2009. 39

[HMG+03] Hugo Haas, Nilo Mitra, Martin Gudgin, Marc Hadley, Noah Mendelsohn, Jean-Jacques Moreau, e Henrik Frystyk Nielsen. W3C SOAP 1.2 Recommendation, 2003. 6

[HWTS07] San-Yih Hwang, Haojun Wang, Jian Tang, e Jaideep Srivastava. A probabilistic approach to modeling and estimating the QoS of web-services-based workflows. *Information Sciences*, 177(23):5484–5503, 2007. 45

[IBM06] IBM Corporation. IBM Web Services, 2006. 9

[ISO86] ISO. SGML. Standard Generalized Markup Language, ISO 8879. 1986., 1986. 6

[ISO00] ISO. HTML. HyperText Markup Language, ISO 15445., 2000. 6

[JDM10] Yassine Jamoussi, Maha Driss, e La Manouba. QoS Assurance for Service-Based Applications Using Discrete-Event Simulation. *IJCSI International Journal of Computer Science Issues*, 7(4):11, 2010. 1

[JKB+09] Joseph C Jacob, Daniel S Katz, G Bruce Berriman, John Good, Anastasia C Laity, Ewa Deelman, Carl Kesselman, Gurmeet Singh, Mei-hui Su, Thomas A Prince, e Roy Williams. Montage: a grid portal and software toolkit for science-grade astronomical image mosaicking. *Int. J. Comput. Sci. Eng.*, 4:73–87, 2009. 31

[JMS02] Li-jie Jin, Vijay Machiraju, e Akhil Sahai. Analysis on Service Level Agreement of Web Services. Relatório técnico, Hewlett-Packard Laboratorie, 2002. 42

[JTK00] By Yuming Jiang, Chen-khong Tham, e Chi-chung Ko. Challenges and approaches in providing QoS monitoring. *International Journal of Network Management*, 10(6):323 – 334, 2000. 46

[KELL10] Oliver Kopp, Lasse Engler, Tammo Van Lessen, e Frank Leymann. Interaction Choreography Models in BPEL: Choreographies on the Enterprise Service Bus. Em *SBPM ONE 2010 the Subjectoriented BPM Conference (2010)*, 2010. 2, 29, 55, 57

[KL03] Alexander Keller e Heiko Ludwig. The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services. *Journal of Network and Systems Management*, 11(1):57–81, 2003. viii, 43, 60

[KLN08] Oliver Kopp, Tammo Van Lessen, e Jörg Nitzsche. The Need for a Choreography-aware Service Bus. *YR-SOC 2008*, (Yrsoc), 2008. 29

[KLW10] Oliver Kopp, Frank Leymann, e Fei Wu. Mapping Interconnection Choreography Models to Interaction Choreography Models. Em *2nd Central-European Workshop on Services and their Composition, Services und ihre Komposition, ZEUS 2010.*, volume 563 of *CEUR Workshop Proceedings*, páginas 81–88, 2010. viii, 27, 28, 30

[KLW11] Oliver Kopp, Frank Leymann, e Sebastian Wagner. Modeling Choreographies: BPMN 2.0 versus BPEL-based Approaches. *Business*, 2011. 26, 27

[Kum09] Bhavish Kumar. Getting Serious about Enterprise Architecture. Relatório técnico, JBoss Savara Project, 2009. 24

[KWH07] Zuling Kang, Hongbing Wang, e Patrick C.K. Hung. WS-CDL+: An Extended WS-CDL Execution Engine for Web Service Collaboration. Em *Proceedings of the IEEE International Conference on Web Services (ICWS 2007)*, páginas 928–935. IEEE Computer Society, 2007. 24, 29, 56

[Lie11] Benedikt Liegener. Service Choreography - Definitions, 2011. 17

[LK08] Niels Lohmann e Oliver Kopp. Tool Support for BPEL4Chor Choreographies. Em Monika Solanki, Barry Norton, e Stephan Reiff-Marganiec, editors, *Proceedings of the 3rd Young Researchers Workshop on Service Oriented Computing YR-SOC*, páginas 74–75, London, 2008. 24

[LLM+10] An Liu, Qing Li, Senior Member, Liusheng Huang, e Mingjun Xiao. FACTS: A Framework for Fault-Tolerant Composition of Transactional Web Services. *Framework*, 3(1):46–59, 2010. 54

[LW10] Niels Lohmann e Karsten Wolf. Realizability is controllability. Em *Proceeedings of the 6th International Workshop on Web Services and Formal Methods*, WS-FM'09, páginas 110–127, Bologna, Italy, 2010. Springer-Verlag. 18

[MGI09] Nebil Ben Mabrouk, Nikolaos Georgantas, e Valerie Issarny. A Semantic End-to-End QoS Model for Dynamic Service Oriented Environments. *Business*, (i):34–41, 2009. 39

[MH05] Jan Mendling e Michael Hafner. From Inter-Organizational Workflows to Process Execution: Generating BPEL from WS-CDL. *OTM Workshops 05*, páginas 506—-515, 2005. 20

[ML08] T Matheis e P Loos. Monitoring cross-organizational business processes. Em *International Conference on E-Learning, E-Business, Enterprise Information Systems, and E-Government CSREA EEE*, páginas 270–275, Nevada, USA, 2008. CSREA Press. 47

[Mon03] Paul Monday. *Web Service Patterns: Java Edition*. Apress, 2003. 10

[MPRW10] Michele Mancioppi, Mikhail Perepletchikov, C Ryan, e WJ. Towards a Quality Model for Choreography. Em *Proceedings of the 2009 International Conference on Service-Oriented Computing*, ICSOC/ServiceWave'09, páginas 435–444, Stockholm, Sweden, 2010. Springer-Verlag. 21, 22

[MRLD09] Anton Michlmayr, Florian Rosenberg, Philipp Leitner, e Schahram Dustdar. Comprehensive QoS monitoring of Web services and event-based SLA violation detection. *Proceedings of the 4th International Workshop on Middleware for Service Oriented Computing - MWSOC '09*, páginas 1–6, 2009. viii, 36, 37, 42, 43, 46

[MSDC12] Daniele Miorandi, Sabrina Sicari, Francesco De Pellegrini, e Imrich Chlamtac. Internet of things: Vision, applications and research challenges. *Ad Hoc Networks*, 10(7):1497–1516, Setembro 2012. 14, 15

[New02] Eric Newcomer. *Understanding Web Services: XML, WSDL, SOAP, and UDDI*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 2002. 4

[NF10] O M G Document Number e P D F Associated File. BPMN 2.0 by Example. 2010. viii, 28, 29

[OAS05] OASIS Standard. OASIS ebXML (Electronic Business using eXtensible Markup Language), 2005. 19

[OLAR09] Cesar Oliveira, Ricardo Lima, Thiago Andre, e Hajo A Reijers. Modeling and Analyzing Resource-Constrained Business Processes. Em *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on*, páginas 2824 –2830. IEEE Computer Society, 2009. 54

[OMG11] OMG. Documents Associated With Business Process Model And Notation (BPMN) Version 2.0, 2011. 26

[Pan10] Ravi Shankar Pandey. A Meta-Model Based Proposal for QOS of WSCDL Choreography. Em *Proceeedings of the International MultiConference of Engineering and Computer Scientists IMECS 2010*, volume I, páginas 2–8, Hong Kong, 2010. Newswood Limited. viii, 41, 46, 47

[PC08] Ravi Shankar Pandey e B D Chaudhary. A Cost Model for Participating Roles Based on Choreography Semantics. *Proceedings of the IEEE Asia-Pacific Services Computing Conference*, páginas 277–283, 2008. 45, 47

[PDKL08] Kerstin Pfitzner, Gero Decker, Oliver Kopp, e Frank Leymann. Web Service Choreography Configurations for BPMN. *Architecture*, 2008. 26

[Pel03] C. Peltz. Web services orchestration and choreography. *Computer*, 36(10):46–52, 2003. viii, 20

[Pel05] Vicente Pelechano. Web Services Standards, Extensions and Future Prospects. Relatório técnico, Department of Information Systems, University of Valencia, Valencia, 2005. viii, 6, 7, 8

[PH07] Mike P. Papazoglou e Willem-Jan Heuvel. Service oriented architectures: approaches, technologies and research issues. *The VLDB Journal*, 16(3):389–415, Março 2007. 8

[PML09] Yu-Yen Peng, Shang-Pin Ma, e Jonathan Lee. REST2SOAP: A framework to integrate SOAP services and RESTful services. Em *Proceedings of the IEEE International Conference on Service-Oriented Computing and Applications (SOCA)*, páginas 1–4. IEEE Computer Society, 2009. 6

[PSW+07] Geguang Pu, Jianqi Shi, Zheng Wang, Lu Jin, Jing Liu, e Jifeng He. The Validation and Verification of WSCDL. *14th Asia-Pacific Software Engineering Conference (APSEC'07)*, (60603033):81–88, Dezembro 2007. 23

[PTDL07] Michael P Papazoglou, Paolo Traverso, Schahram Dustdar, e Frank Leymann. Service-Oriented Computing: State of the Art and Research Challenges. *Computer*, 40(11):38–45, 2007. viii, 8, 9, 14

[PTDL08] Michael P. Papazoglou, Paolo Traverso, Schahram Dustdar, e Frank Leymann. Service-Oriented Computing: a Research Roadmap. *International Journal of Cooperative Information Systems*, 17(02):223, 2008. 1, 8

[RBHJ08] Sidney Rosario, Albert Benveniste, Stefan Haar, e Claude Jard. Probabilistic QoS and Soft Contracts for Transaction-Based Web Services Orchestrations. *IEEE Transactions on Services Computing*, 1(4):187–200, 2008. viii, x, 44, 45, 60

[RBJ09] Sidney Rosario, Albert Benveniste, e Claude Jard. Monitoring probabilistic SLAs in Web service orchestrations. Em *Proceedings of the 11th IFIP/IEEE international conference on Symposium on Integrated Network Management*, IM'09, páginas 474–481, iscataway, NJ, USA, 2009. IEEE Press. 45, 48, 60

[REM+07] Florian Rosenberg, Christian Enzi, Anton Michlmayr, Christian Platzer, e Schahram Dustdar. Integrating Quality of Service Aspects in Top-Down Business Process Development Using WS-CDL and WS-BPEL. *Enterprise Distributed Object Computing Conference, IEEE International*, 0:15, 2007. 42, 43

[Rob05] David Robertson. A Lightweight Coordination Calculus for Agent Systems. Em *Declarative Agent Languages and Technologies II Second International Workshop, DALT 2004, Revised Selected Papers*, páginas 183–197. Springer Berlin / Heidelberg, New York - USA, 2005. 24

[Ros09] Florian Rosenberg. *QoS-Aware Composition of Adaptive Service-Oriented Systems*. Tese de Doutorado, University Vienna - Austria, 2009. viii, 1, 19, 36, 40, 42, 46, 48

[RR09] Michael Von Riegen e Norbert Ritter. Reliable Monitoring for Runtime Validation of Choreographies. Em *Proceedings of the 2009 Fourth International Conference on Internet and Web Applications and Services*, páginas 310–315. IEEE Computer Society, 2009. 46

[Rt05] Stephen Ross-talbot. Orchestration and Choreography: Standards, Tools and Technologies for Distributed Workflows. *Workshop on Network Tools and Applications in Biology (NETTLAB 2005)*, página 8, 2005. 17, 19

[SAV09] SAVARA Project. Savara - Project Charter. Relatório Técnico August, Red Hat Ltd, 2009. 24

[SBFZ07] Jianwen Su, Tevfik Bultan, Xiang Fu, e Xiangpeng Zhao. Towards a theory of web service choreographies. Em *Proceedings of the 4th international conference on Web services and formal methods*, WS-FM'07, páginas 1–16, Brisbane, Australia, 2007. Springer-Verlag. 17, 18, 21, 22, 23

[SL11] Richard Simard e Pierre L'Ecuyer. Journal of Statistical Software. *Journal Of Statistical Software*, 39(11), 2011. 62

[SLD+10] Jun Sun, Yang Liu, Jin Song Dong, Geguang Pu, e Tian Huat Tan. Model-based Methods for Linking Web Service Choreography and Orchestration. Em *Proceedings of the Asia Pacific Software Engineering Conference APSEC '10*, páginas 166—-175. IEEE Computer Society, 2010. 21

[SR09] G Salaün e N Roohi. On Realizability and Dynamic Reconfiguration of Choreographies. Em *Proceeedings of the Workshops on Autonomic and Self-Adaptative Systems WASELF '09*, volume 3, páginas 21–31. JISBD 2009, 2009. 22

[SSDS10] M Sathya, M Swarnamugi, P Dhavachelvan, e G Sureshkumar. Evaluation of QoS Based Web- Service Selection Techniques for Service Composition. *International Journal of Software Engineering (IJSE)*, (1):73–90, 2010. 36

[Stu12] U Stuttgart. Research Challenges on Adaptive Software and Services in the Future Internet: Towards an S-Cube Research Roadmap. Em *Software Services and Systems Research - Results and Challenges (S-Cube), 2012 Workshop on European*, páginas 1–7, 2012. 1

[TAM00] E.S.H. Tse-Au e P.a. Morreale. End-to-end QoS measurement: analytic methodology of application response time vs. tunable latency in IP networks. Em *Proceedings of the IEEE/IFIP Network Operations and Management Symposium 'The Networked Planet: Management Beyond 2000' (Cat. No.00CB37074)*, páginas 129–142, Honolulu, HI, USA, 2000. IEEE Computer Society. 48

[TPM06] Otón Salvador Tortosa, Lourdes Jiménez Rodríguez Roberto Barchino Plata, e José Ramón Hilera González José María Gutiérrez Martínez. Web Services Oriented Architecture for the Implementation of Learning Objects Distributed Repositories. Em Nuno Guimarães, Pedro Isaías, e Ambrosio Goikoetxea, editors, *Proceedings of the IADIS International Conference on Applied Computing*, página 313. Alcala, IADIS, 2006. 4, 5

[TW10] Lu Tan e Neng Wang. Future Internet: The Internet of Things. Em *Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on*, volume 5, páginas V5–376 –V5–380, 2010. viii, 14, 16

[UDD05] UDDI OASIS Standard. UDDI, 2005. 6

[UHS11] Irfan Ul Haq, Altaf Ahmad Huqqani, e Erich Schikuta. Hierarchical aggregation of Service Level Agreements. *Data & Knowledge Engineering*, 70(5):435–447, Maio 2011. viii, 34

[UPSB10] Irfan Ul Haq, Adrian Paschke, Erich Schikuta, e Harold Boley. Rule-based validation of SLA choreographies. *The Journal of Supercomputing*, 9999:1—-22, 2010. 47

[Van03] A. P. Van Der Aalst, W. M. P. and Ter Hofstede, A. H. M. and Kiepuszewski, B. and Barros. Workflow Patterns. *Distributed and Parallel Databases*, 14(1):5–51, 2003. 42

[VGF10] Hugues Vincent, Nikolaos Georgantas, e Emilio Francesquini. CHOReOS : Scaling Choreographies for the Internet of the Future. Em *Middleware '10 Posters and Demos Trac*, number i in Middleware Posters '10, páginas 8:1—-8:3, Bangalore, India, 2010. ACM. 1

[W3C98] W3C Consortium. W3C Working Group. eXtensible Markup Language (XML), 1998. 6, 9

[W3C05] W3C Candidate Recommendation. Web Services Choreography Description Language, version1.0, 2005. 18

[Wan04] H Wang. Web services: problems and future directions. *Web Semantics: Science, Services and Agents on the World Wide Web*, 1(3):309–320, Abril 2004. 5

[Web05] Web Services Choreography Working Group. Web Services Choreography Description Language Version 1.0, 2005. 24

[Win99] D Winer. XML-RPC Specification, 1999. 9

[WKK+10] Branimir Wetzstein, Dimka Karastoyanova, Oliver Kopp, Frank Leymann, e Daniel Zwink. Cross-organizational process monitoring based on service choreographies. Em *Proceedings of the 2010 ACM Symposium on Applied Computing - SAC '10*, página 2485, New York, New York, USA, 2010. ACM Press. 47

[WSI05] WSI. Web Services Interoperability Organization, 2005. 4

[XCHZ07] Zhao Xiangpeng, Cai Chao, Yang Hongli, e Qiu Zongyan. A QoS View of Web Service Choreography. *IEEE International Conference on e-Business Engineering (ICEBE'07)*, páginas 607–611, Outubro 2007. 45, 47

[XCZH09] Yunni Xia, Jun Chen, Mingqiang Zhou, e Yu Huang. A Petri-Net-Based Approach to QoS Estimation of Web Service Choreographies. Em *Advances in Web and Network Technologies, and Information Management*, number 2006, páginas 113–124. Springer-Verlag, Berlin, Heidelberg, 2009. 45, 48

[YZQ+06] Hongli Yang, Xiangpeng Zhao, Zongyan Qiu, Geguang Pu, e Shuling Wang. A Formal Model for Web Service Choreography Description Language (WS-CDL). Em *Proceedings of the IEEE International Conference on Web Services (ICWS'06)*, number 605730081, páginas 893–894. IEEE Computer Society, 2006. 45

[ZBDH06] Johannes Maria Zaha, Alistair Barros, Marlon Dumas, e Arthur Hofstede. Letâs Dance: A Language for Service Behavior Modeling. Em *OTM Proceedings of the 14th International Conference on Cooperative Information Systems (CoopIS'06)*, Lecture Notes in Computer Science, páginas 145–162. Springer, 2006. 18, 22, 23

[ZTW+06] X. Zhou, W. Tsai, X. Wei, Y. Chen, e B. Xiao. Pi4SOA: A Policy Infrastructure for Verification and Control of Service Collaboration. *Proceedings of the IEEE International Conference on e-Business Engineering (ICEBE'06)*, páginas 307–314, 2006. 24, 55, 57

[ZWPZ10] Yongxin Zhao, Zheng Wang, Geguang Pu, e Huibiao Zhu. A Formal Model for Service Choreography with Exception Handling and Finalization. Em *Theoretical Aspects of Software Engineering (TASE), 2010 4th IEEE International Symposium on*, páginas 15–24, 2010. 18, 21

[ZXCH07] Qiu Zongyan, Zhao Xiangpeng, Cai Chao, e Yang Hongli. Towards the Theoretical Foundation of Choreography. Em *Proceedings of the 16th international conference on World Wide Web*, number 60573081 in WWW '07, Banff, Alberta, Canada, 2007. ACM. 23, 24

[ZYZ10] Huiyuan Zheng, Jian Yang, e Weiliang Zhao. QoS Probability Distribution Estimation for Web Services and Service Compositions. Em *Proceedings of the EEE International Conference on Service-Oriented Computing and Applications, SOCA '10*, páginas 1—-8, Perth, Australia, 2010. IEEE Computer Society. 44, 45, 49